NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

NUMERICAL OPTIMIZATION USING DESKTOP COMPUTERS

by

Walter Bacon Cole

September 1980

Thesis Advisors: G. N. Vanderplaats
M. D. Kelleher

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Numerical Optimization Using Desktop Computers		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; September 1980
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Walter Bacon Cole		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE September 1980
		13. NUMBER OF PAGES 164 pages
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Numerical optimization, desktop computers, energy conversion, nonimaging solar collectors

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Two computer programs were developed in advanced BASIC to perform numerical optimization of a user supplied design problem on the Hewlett Packard 9845A desktop computer. An executive program, OPCON, provides the interactive link between the computer user and the DESOP numberical optimization program. DESOP performs the numerical optimization using the sequential unconstrained minimization technique with an external penalty function. The unconstrained subproblem is solved using the Fletcher-Reeves method of

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
(Page 1) S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

conjugate directions, and using Golden Section search and polynomial interpolation in the one-dimensional search.

A computer subprogram, NISCO, was developed in advanced BASIC to model a nonimaging concentrating compound parabolic trough solar collector. Thermophysical, geophysical, optical and economic analyses were used to compute a life-cycle fuel savings, for a design of stated thermal capacity. NISCO was coupled to the OPCON/DESOP optimization program to find the design which maximizes the life-cycle fuel savings.

Approved for public release; distribution unlimited

Numerical Optimization Using Desktop Computers

by

Walter Bacon Cole
Lieutenant, United States Navy
B.S.M.E., Purdue University, 1974

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 1980

ABSTRACT

Two computer programs were developed in advanced BASIC to perform numerical optimization of a user supplied design problem on the Hewlett Packard 9845A desktop computer. An executive program, OPCON, provides the interactive link between the computer user and the DESOP numerical optimization program. DESOP performs the numerical optimization using the sequential unconstrained minimization technique with an external penalty function. The unconstrained subproblem is solved using the Fletcher-Reeves method of conjugate directions, and using Golden Section search and polynomial interpolation in the one-dimensional search.

A computer subprogram, NISCO, was developed in advanced BASIC to model a nonimaging concentrating compound parabolic trough solar collector. Thermophysical, geophysical, optical and economic analyses were used to compute a life-cycle fuel savings, for a design of stated thermal capacity. NISCO was coupled to the OPCON/DESOP optimization program to find the design which maximizes the life-cycle fuel savings.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

English Letter Symbols

A_r	- Solar collector receiver surface area
A_t	- One half the solar collector aperature area
B	- Cantilevered beam width
C_d	- Solar collector depth
C_p	- Specific heat
CR	- Solar collector concentration ratio
E	- Young's modulus of elasticity, or the penalty function exponent used in DESOP
F_1	- Lower Golden Section fraction
F_2	- Upper Golden Section fraction
H	- Cantilevered beam height
$\bar{H}$	- Equality constraint vector
Icalc	- A user's flag in the DESOP program for initial and final user generated output
L	- Cantilevered beam length, or the solar collector length
$\dot{m}$	- Solar collector mass flow rate
Mfr	- Solar collector mass flow rate
Ndv	- Number of design variables
Obj	- Objective function
Opj	- Penalized objective function
P	- Cantilevered beam load
Qa	- Solar collector heat available

Qu	- Solar collector heat gain
Qy	- Yearly solar collector heat available
q_i	- Heat flux
R	- Penalty parameter used in DESOP or the solar collector receiver radius
r	- Solar collector receiver radius
Tap	- Solar collector aperature cover temperature
Tc2	- Solar collector coolant exit temperature
Thetai	- Solar collector acceptance half angle
Thetat	- Solar collector truncation angle
t	- Solar collector distance between the reflector and a point tangent to the receiver
V	- Cantilevered beam volume
x	- Solar collector reflector coordinate
$\bar{X}$	- Design variable vector
y	- Solar collector reflector coordinate

Greek Letter Symbols

α	- One-dimensional search move parameter
δ	- Cantilevered beam deflection
Θ	- Solar collector geometry angle measured from the collector centerline (See Fig. 6)
Θ_i	- Solar collector acceptance half angle
Θ_t	- Solar collector truncation angle
ν	- Cantilevered beam shear stress
σ_b	- Cantilevered beam bending stress

ACKNOWLEDGEMENT

The author gratefully acknowledges the aid he has received from several sources. He is indebted to his thesis advisors, Professors G. N. Vanderplaats and M. D. Kelleher, for their guidance and support throughout this project. The author also wishes to thank his wife for her understanding and encouragement during this period.

I. INTRODUCTION

A. BACKGROUND

Most engineering design problems contain several continuous variables and as such have an infinite number of solutions. The purpose of optimization is to find the best possible solution among the many potential solutions for a given problem in terms of some effectiveness or performance criteria. There are several methods of optimization. The methods may be classified as follows:

Analytical methods which use the classical techniques of differential calculus and the calculus of variations.

Numerical methods which use past information to generate better solutions to the optimization problem by means of iterative procedures. Numerical methods can be used to solve problems that cannot be solved analytically.

Graphical methods which use the preparation of a plot of the parameter to be optimized as a function of one or more variables. This method although simple and easy to use becomes unmanageable when there are three or more design variables.

Experimental methods which use direct experimentation of the actual process, the results of one experiment being used to decide on where to perform the next experiment.

Case study methods which evaluate the results from a number of representative cases, and choose the "best" solution. The "best" solution is thus not likely to be the optimum solution.

Of the optimization methods, the numerical method lends itself to computerized solution. As design is an interactive process between the designer and the problem, and the desktop computer lends itself towards dedicated interactive use, the development of a numerical optimization program for use on a desktop computer in an interactive mode, is the objective of this thesis.

To date a great amount of effort has been spent developing reliable and efficient optimization programs for mainframe computers. These programs are fairly large and complex, requiring a substantial amount of core space during execution. The size and complexity of the programs has been the result of an attempt to minimize the amount of computer time required to perform an optimization and thus the cost to the user. With the advent and availability of desktop computers, there has been a sharp reduction in the cost of computer time to the computer user. While the desktop computer has far less core space than a mainframe computer, once the time factor is removed from the numerical optimization process it is possible to put a small but reliable numerical optimization program on a desktop computer. A design problem concerning the optimal geometry

for a nonimaging parabolic trough solar collector was developed to demonstrate the engineering application of the numerical optimization program developed for this thesis.

B. SCOPE

The numerical optimization of a given function may be accomplished using many varied and different algorithms. Some of the more popular methods used are: random search, linear programming, feasible directions, Golden Section, Newton's method and sequential unconstrained minimization. A particular optimization program will use one or more of these methods to efficiently and reliably arrive at the best solution to a particular problem.

For this thesis two computer programs were developed to perform numerical optimization on a desktop computer. The first program was developed as an executive program to control the optimization process. The executive program is named OPCON which stands for OPtimizer CONtrol program. OPCON provides the interactive link between the user and the program which performs the numerical optimization. OPCON allows the user to input data, attach a specified analysis subprogram to the numerical optimization program and control execution of the numerical optimization program. The second program developed was the numerical optimization program, DESOP. DESOP stands for DEsktop Sequential unconstrained minimization technique Optimization Program.

DESOP performs the numerical optimization of the user supplied problem using the sequential unconstrained minimization technique with an external penalty function. The unconstrained subproblem is solved using the Fletcher-Reeves method of conjugate directions, Golden Section search, and polynomial interpolation.

The third computer program, NISCO, was developed to model a nonimaging concentrating compound parabolic trough solar collector using thermophysical, geophysical, optical and economic analysis to compute a life-cycle cost for a design with a stated energy capacity. NISCO stands for NonImaging concentrating compound parabolic trough Solar COllector.

C. OBJECTIVE

The objective of this thesis was to develop a system of interactive programs for the Hewlett-Packard 9845A desktop computer which perform numerical optimization, and to demonstrate the capability on the design of a nonimaging concentrating compound parabolic trough solar collector. Three programs were developed to meet the objective: an executive program, a numerical optimization program and a solar collector analysis program.

The purposes of the executive program OPCON are:

1. To provide a primary point of contact for the computer user from which to effect a numerical optimization on any number of user prepared analysis subprograms.

2. To provide a standardized, formatted input for the design variables, side constraints and optimizer control parameters, which is recognizable by all the programs in the numerical optimization package of programs.

3. To control the operation of the different optimization and design analysis programs within the system through a process of program overlays which maximizes the computer space available for the design analysis program.

4. To develop a program which is portable to different computers using an advanced BASIC language.

The purposes of the numerical optimization program DESOP are to develop an optimization program that:

1. Is relialbe in reaching a design optimum, irrespective of the starting point.

2. Will arrive near the design optimum using default values for the optimizer control variables.

3. Will allow the user to monitor the optimization process and to change the optimizer control variables to more efficiently and/or more accurately reach the design optimum.

4. Is portable to different computers using an advanced BASIC language.

The purposes of the solar collector program NISCO are to:

1. Model a nonimaging concentrating compound parabolic trough solar collector using a system of thermophysical, geophysical, optical and economic equations.

2. Arrive at an optimum design for a solar collector given a stated average daily heat gain and a life-cycle period.

II. NUMERICAL OPTIMIZATION

A. THE CONCEPT OF NUMERICAL OPTIMIZATION

Consider the problem of designing the cantilevered beam shown in Figure 1. The design task may be broken down into three major parts. First, the objective of the design must be determined, which in this case is to minimize the weight of the beam required to support the concentrated tip load P. Second, any physical constraints that may effect the design must be determined. Thirdly, any limits which exist on the design variables must be determined. The design problem may then be reduced to a system of equations as follows:

Minimize the volume (V)

$$V = B \cdot H \cdot L$$

. Subject to:

Bending stress (σ_b)

$$\sigma_b = \frac{6 \cdot P \cdot L}{B \cdot H^2} \leq 20000 \text{ psi}$$

Shear stress (ν)

$$\nu = \frac{3 \cdot P}{2 \cdot B \cdot H} \leq 10000 \text{ psi}$$

Deflection under load (δ)

$$\delta = \frac{4 \cdot P \cdot L^3}{E \cdot B \cdot H^3} \leq 1 \text{ inch}$$

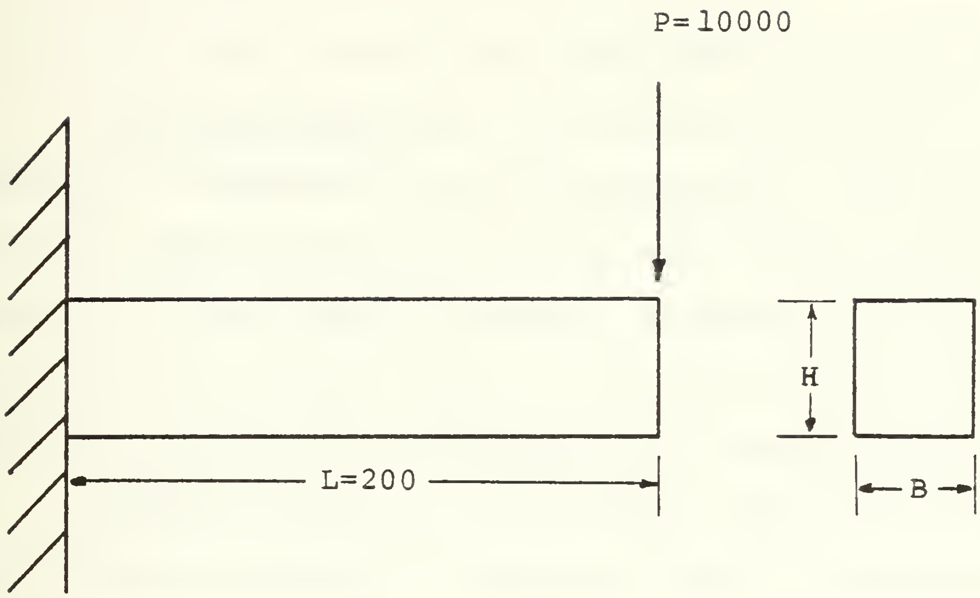


Figure 1. Cantilevered Beam Design Problem

With geometric constraints such that:

$$0.5 \leq B \leq 0.5$$

$$1.0 \leq H \leq 20.0$$

$$H/B \leq 10.0$$

At this time the following definitions are introduced:

Objective Function - The parameter which is to be minimized or maximized during optimization. The objective function always occurs on the left side of the equation unless it is also used as a design variable. An objective function may be either linear or nonlinear, implicit or explicit, but must be a function of the design variables.

Design Variable - Any parameter which the optimization process is allowed to change in order to improve the design. Design variables appear only on the right hand side of equations in the analysis program.

Inequality Constraint - Any parameter which must not exceed specified bounds for the design to be acceptable. Constraint functions always appear on the left side of equations. A constraint may be linear, nonlinear, implicit or explicit, but must be a function of one or more design variables.

Equality Constraint - Any parameter which must equal a specified value for the design to be acceptable. The same rules apply to equality constraints as inequality constraints.

Side Constraint - Any upper or lower bound placed upon a design variable. Side constraints are usually not included in the system of equations that comprise an analysis program. Instead they are usually included as part of the data input to the optimization program.

Analysis Code - The system of equations utilizing the design variables which are used to calculate the objective function and the constraints of a particular design problem.

The general optimization problem may thus be stated mathematically as:

Find the set of design variables $\bar{X}_i$ where $i = 1,2,\ldots,n$ which will:

Minimize the objective function (Obj)

$$Obj = f(\bar{X})$$

Subject to:

Inequality constraints (G)

$$G_j(\bar{X}) \leq 0 \qquad j = 1,2,\ldots,m$$

Equality constraints (H)

$$H_j(\bar{X}) = 0 \qquad j = 1,2,\ldots,l$$

Side constraints

$$X_i^l \leq X_i \leq X_i^u \qquad i = 1,2,\ldots,n$$

Returning to the cantilevered beam problem, it may be stated in the standard format as follows:

Let $X(1) = B$, $X(2) = H$, and $Obj = Vol = B \cdot H \cdot L$

Then minimize $Obj = Vol$

Subject to:

$$G(1) = \frac{\sigma_b}{20000} - 1 \leq 0$$

$$G(2) = \frac{\tau}{10000} - 1 \leq 0$$

$$G(3) = \delta - 1 \leq 0$$

$$G(4) = \frac{H}{B} - 10 \leq 0$$

With side constraints:

$$X(1)^l = 0.5$$

$$X(1)^u = 5.0$$

$$X(2)^l = 1.0$$

$$X(2)^u = 20.0$$

It is thus fairly simple and straightforward to perform an analysis on a particular beam for a given B and H. Successive analyses may be performed on the cantilevered beam by solving the above system of equations. It is desirable to automate the successive solutions and to direct the solutions such that each solution is a better design than the last. One approach for doing so, and the one used by DESOP is to proceed as follows: Start with initial values for B and H. Solve the above set of equations to find the objective function Obj and to see if any constraints are violated. A pseudo objective function is created to represent designs when constraints are violated. If a constraint is violated, a penalty is added to Obj to form a penalized objective function Opj. The gradient of the

penalized objective function at the initial design may be found by taking the first partial derivative of Opj with respect to the design variables. The gradient of the penalized objective function defines the direction of steepest ascent. In the case of the cantilevered beam, it is desired to minimize the objective function; therefore, the greatest improvement in design may be achieved by moving in the negative gradient, or steepest descent direction. From the initial design point a search is performed in the steepest descent direction for the minimum value of Opj in that direction. At the new minimum, the gradient of the penalized objective function is again determined and a search is performed in a conjugate direction until a second minimum is found. Successive iterations are performed until the gradient is found to be zero or each successive iteration produces a sufficiently small change in Opj such that for all practical purposes the minimum has been found. At this time the penalty function is increased. If the design is in a region where there are no constraints violated an increase in the penalty function will not change the value of Opj. If on the other hand the design is in an infeasible region where there are one or more constraints violated, Opj will be increased, and the search for a new minimum will commence. If the minimum of the objective function exists in the infeasible region,

the minimum value for the objective function in the feasible region will be approached from the infeasible region as the penalty function is increased. The design improvement process will terminate when a zero gradient is found or successive iterations produce a sufficiently small change in the value of Opj and an increase in the penalty function causes no change in Opj.

The minimum thus reached by the optimization process is a minimum with respect to the penalized objective surface immediately surrounding the final design point. The optimization process cannot distinguish between local and global minimum points. It is thus good engineering practice to run several optimizations for a particular design problem from several different initial design points. If optimizations performed from different initial design points converge on the same minimum point, that point is probably a global minimum. If on the other hand two or more minimums are found, there may be local minimums located in the design space being considered and care must be taken to find the global minimum.

B. THE DESOP NUMERICAL OPTIMIZATION PROGRAM

The DEsktop Sequential unconstrained minimization technique Optimization Program was developed using the basic optimization approach outlined in Section II.A. A copy of the program is included in Appendix E. The major

program structure is shown in Figure 2. The following discussion will refer to Figure 2 and describe the major features of the program.

1. Basic Program Execution

The DESOP program begins execution when it is loaded, linked to the user's analysis subprogram, and the program is instructed to run by the OPCON program. The above actions are automatically performed by the OPCON program. DESOP is loaded into the computer by an overlay process. Therefore no variables can be directly transferred between the DESOP and OPCON programs. DESOP begins execution by reading the optimizer control variables and the design variables that were input using the OPCON program and saved to a mass storage device. The program then sets Icalc equal to one and evaluates the objective function and constraints at the initial design point. Icalc is a flag provided the user to key user specified output on the initial and final design analysis. DESOP will provide the user with a hard copy output of the initial design variables, the value of the constraints, the objective function and the penalized objective function. The user then has the option of continuing with the DESOP program to optimize his analysis subprogram or to return to the OPCON program to change one or more of the input parameters.

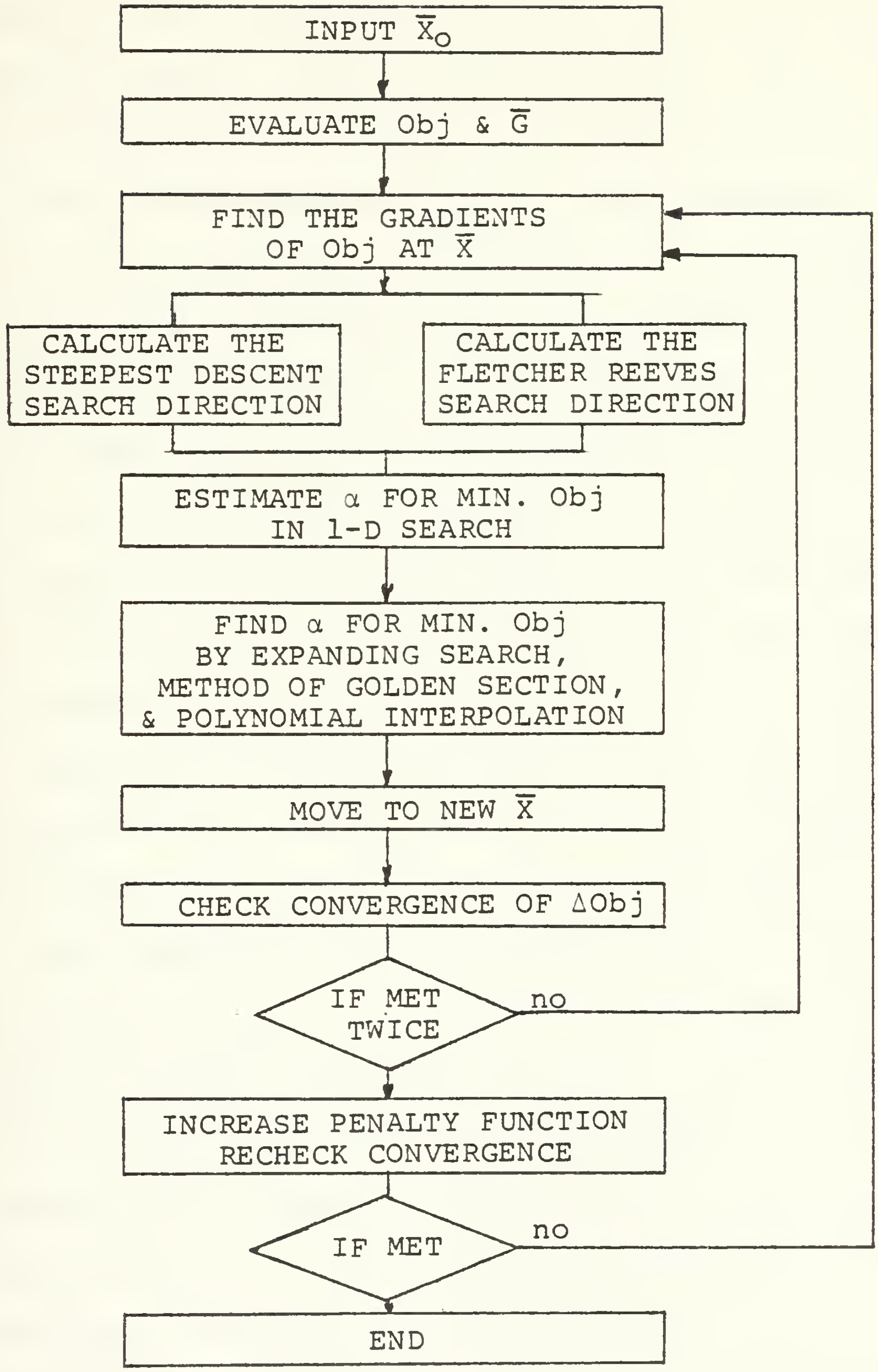


Figure 2. DESOP Flow Diagram

Proceeding with the optimization, there are two major loops in the optimization program. The outer loop increases the penalty function when the inner loop's convergence criteria have been met. A convergence test is then performed by the outer loop. If the convergence criteria is met, the optimization process is considered finished. If the convergence criteria for the outer loop is not met, program execution is returned to the inner loop. The inner loop performs successive iterations searching for the minimum of the penalized objective function with no increase in the penalty function taking place. When the inner loop's convergence criteria have been met program execution is transferred to the outer loop.

Execution of the program while in the inner loop proceeds as follows: First, the gradient of the penalized objective function is calculated by subroutine GRAD. The program then computes a search direction using either a steepest descent method or the method of conjugate directions developed by Fletcher and Reeves [Ref. 1]. Once a search direction is established the optimizer attempts to locate the minimum value of the penalized objective function in the search direction. This process if referred to as the one-dimensional search and is illustrated in Figure 3. The efficiency and accuracy to which the one-dimensional search for the minimum of the penalized objective function

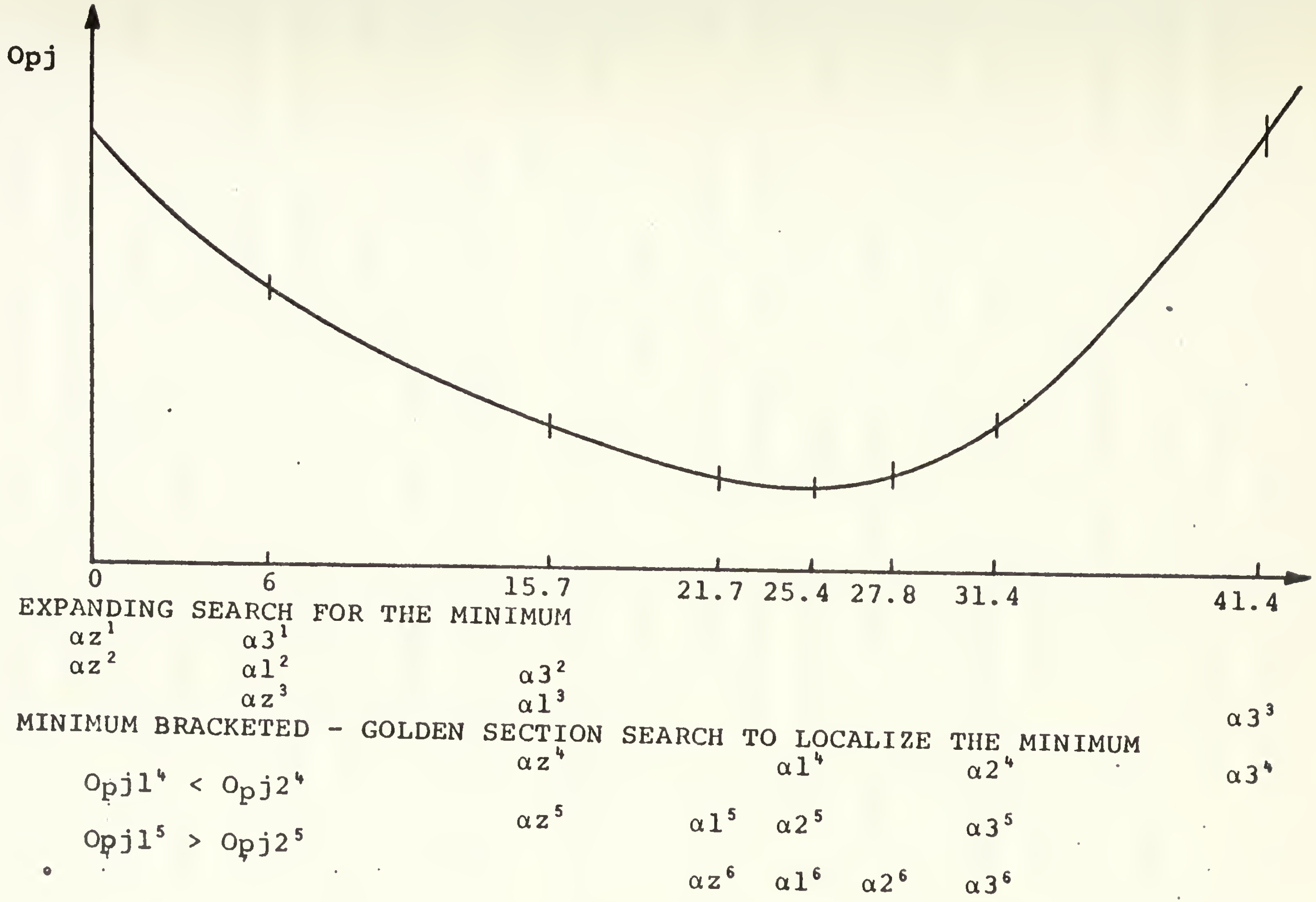


Figure 3. The One-Dimensional Search Process

is accomplished, is the key to successful sequential unconstrained minimization technique numerical optimization. The one-dimensional problem may be expressed in terms of the penalized objective function, Opj, and the amount of movement, α, in the search direction, $\bar{S}$. First the slope of Opj with respect to α is calculated. An initial "guess" of how far to move is made using subroutine ALPGES. The α which corresponds to the minimum value of Opj in the one-dimensional search is then calculated using subroutine ALPBND and subroutine QUEBIC. The minimum value of Opj thus found is then compared to the previous value of Opj for convergence using subroutine CONVRG. If convergence is not met, execution returns to the start of the inner loop. If convergence is met, execution returns to the outer loop.

When the convergence criteria have been met for both the inner and outer loops, the program proceeds to set Icalc to three as a flag for user generated output for the final design. DESOP then provides the user with a hard copy output of the final design variables, objective function value, penalized objective value, constraint values, the number of inner loop iterations, the number of times the analysis subprogram was called and the final value of the penalty function. The OPCON program is then overlayed over the DESOP program and program execution is returned to the OPCON program.

2. Finding the Search Direction

The first step in finding the search direction, $\bar{S}$, is to determine the slope of Opj at the present design point. The forward finite difference method is used.where:

$$\frac{\partial F}{\partial X_i} = \frac{F(X_i + \Delta X_i) - F(X_i)}{\Delta X_i} = -S_i$$

$$i = 1,2,\ldots Ndv$$

As $\partial F/\partial X_i$ gives the direction of positive slope, the search direction is the negative of $\partial F/\partial X_i$. The first search is performed using the steepest descent as found above using the following relation:

$$X_i' = X_i + \alpha S_i$$

where alpha is the distance moved in the $\bar{S}$ direction. When a minimum is obtained along the direction of steepest descent, a new Fletcher-Reeves conjugate search direction [Ref. 1] is calculated at the new Design point using the following relations:

$$S_i' = \frac{\partial F}{\partial X_i} + BS_i$$

$$B = \frac{\sum_{i=1}^{Ndv} [\frac{\partial F'}{\partial X_i}]^2}{\sum_{i=1}^{Ndv} [\frac{\partial F}{X_i}]^2}$$

where the prime denotes values for the present iteration and the non-prime variables indicate values for the previous

iteration. A one-dimensional search is then performed in the new search direction. Searches are continued using the conjugate direction method for Ndv + 1 iterations, where Ndv is the number of design variables. The search process is then restarted using the steepest descent method. The reason for incorporating the conjugate search method is that the steepest descent method when traversing a design surface with a curved valley will tend to zigzag from one side of the design surface valley to the other making very little progress as is illustrated in Figure 4. The conjugate direction method is much more efficient in traversing such a design surface. However, as the conjugate direction method is additive upon previous searches, it has a tendency to decrease in effectiveness with each successive search owing to the accumulation of numerical "noise." For that reason the search process is restarted with the steepest descent method every Ndv + 1 iterations, or when the conjugate direction predicts a positive slope. The search direction is normalized to avoid inaccuracies caused by numerical ill-conditioning.

3. Estimating an Initial Value for Alpha

The initial estimate for alpha is made in the following manner: First, the slope of Opj in the search direction is calculated as the sum of each of the products of the gradients times the search direction. Then the slope of Opj in the search direction is divided by the value

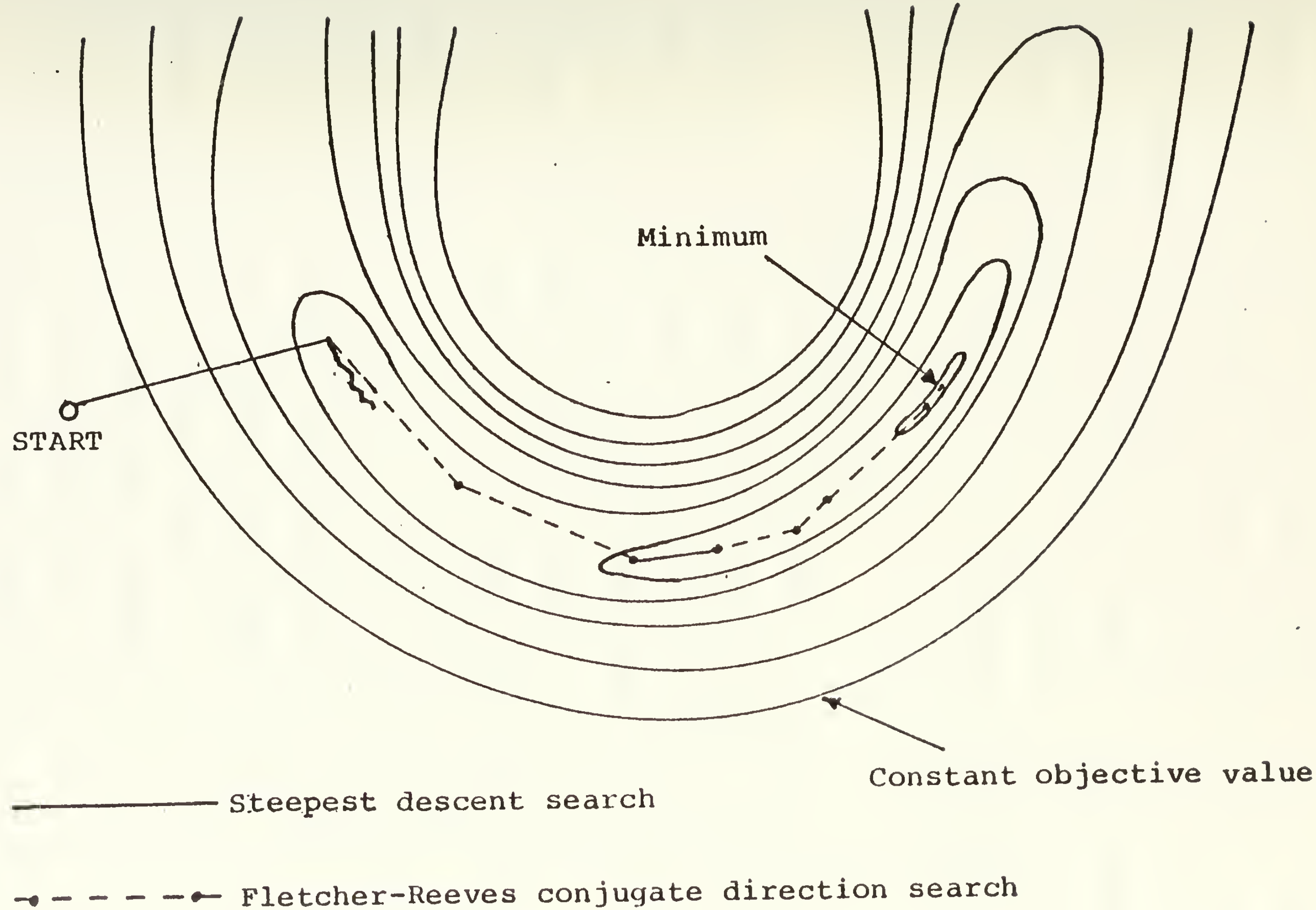


Figure 4. Steepest Descent and Conjugate Direction Search in a Two-Dimension Design Space.

of Opj. This value is then multiplied by an improvement percentage in Opj. This first estimate is then applied to a series of conditional tests to determine the validity of the estimate with respect to the slope of Opj and the magnitude of the design variables. Lastly, the estimate for alpha is checked to see if it violates any side constraints. If it does, the value of the estimate for alpha is reduced until the side constraints are no longer violated.

4. Calculating Alpha

The calculation of alpha is the most critical algorithm in the DESOP program in providing reliable optimizer operation. The ability to accurately and efficiently find the minimum of the penalized objective function in the one-dimensional search affects directly the operation of the optimizer. Figure 5 illustrates the zigzag phenomenon which occurs if alpha is not accurately found. The zigzag phenomenon is caused by the fact that the optimizer in performing the forward finite difference for calculating the search direction perturbs the design vector a very small amount. As such the optimizer can only "see" the design surface that is immediately adjacent to the design point. Therefore, if the minimum is not found in the one-dimensional search, the optimizer will converge very slowly on the minimum.

There are two major sections to the ALPBND subroutine. The first section attempts to find the minimum value of Opj

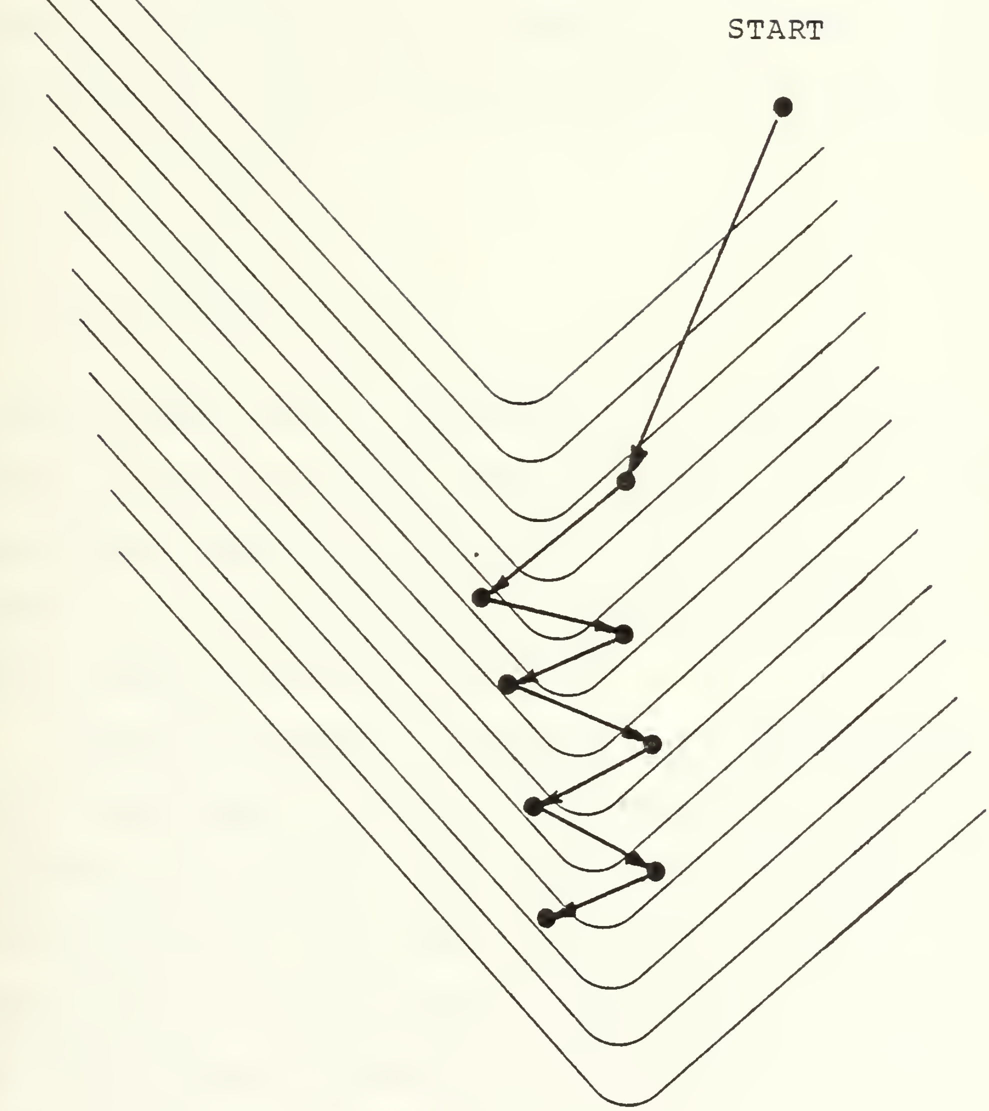


Figure 5. The Zigzag Phenomenon

using an expanding search technique. The first move is the amount predicted by the ALPGES subroutine. If the minimum is not bracketed by the first move, additional moves are made. Each move is larger than the last. The size of the move is increased each time by an amount equal to the size of the last move divided by the lower Golden Section fraction, where the Golden Section fractions are:

$$F_1 = \frac{3 - \sqrt{5}}{2}$$

$$F_2 = \frac{\sqrt{5} - 1}{2}$$

The lower Golden Section fraction, F_1, is used so that the interval will be consistent with the Golden Section search in the second section of the ALPBND subroutine. The expanding search is continued until the minimum value of the objective function has been bracketed.

Once the minimum is bracketed, a Golden Section search is performed to reduce the bracketing interval on the minimum by an amount such that when the two end points of the interval are taken with two points internal to the interval and a cubic is passed through the four points, the cubic will accurately predict the minimum of the penalized objective function. Himmelblau in [Ref. 2] states that the Golden Section search method of reducing the interval around the minimum of Opj is the most effective of the reducing techniques studied. Golden Section search is based

on the splitting of a line into two segments known in ancient times as the "Golden Section." The ratio of the whole line to the larger segment is the same as the ratio of the larger segment to the smaller segment. The two Golden Section fractions are employed to split the interval bracketing the minimum as shown in Figure 3. Once the interval has been split, the two values of Opj corresponding to the internal points are compared to find the larger of the two. The internal point with the larger value of Opj will become the new end point for the interval, the remaining interior point will by the fact that it was determined by a Golden Section fraction, be equal to the point determined by the other Golden Section fraction. Thus, only one new point must be calculated to continue the Golden Section search. The search is continued in this manner until the vertical separation of the two end points with respect to the interior points is less than one percent. The four values of the penalized objective function corresponding to the four Golden Section search points are then sent to a cubic interpolator. The cubic interpolator will return a value for alpha to predict the minimum of the penalized objective function, and the minimum of the cubic function that the interpolator has created. The subroutine ALPBND will then test the predicted minimum with the minimum found at the predicted alpha. If there is less than a tenth of

one percent difference between the two values of the objective function, the point predicted by the cubic interpolator will be accepted as the minimum and program execution will return to the main program. If the predicted minimum is not sufficiently close to the minimum at the predicted alpha, another Golden Section search will be performed to reduce the interval and better localize the minimum. The four points from the reduced interval will then be sent to the cubic interpolation subroutine. This process will continue until either the test for the minimum is positive or the interval has been reduced to less than 1E-12. Program execution will then return to the main program.

5. Subroutine QUEBIC

Subroutine QUEBIC is used to estimate the alpha at which Opj is a minimum based on four point cubic interpolation. If the function more closely resembles a quadratic than a cubic, a three point quadratic interpolation is performed using the three points which bracket the minimum. If the predicted minimum is outside the interval spanned by the two end points again a quadratic interpolation is performed. If the minimum still lies outside the two end points, the analysis returns to subroutine ALPBND, the inverval bracketing the minimum is reduced, and program execution returns to QUEBIC.

6. Convergence of the Penalized Objective Function

The penalized objective function is tested for convergence at the end of each inner loop and again at the end of the outer loop in the main program. Convergence is tested by calling subroutine CONVRG. There are two criteria used for testing for convergence. The first tests the relative difference of the value of Opj from the present iteration with the value of Opj from the last iteration. The second method tests the absolute difference of the two values. The second method is employed for cases when the value of the penalized objective function approaches zero. When convergence has been met on two successive iterations, the penalty function is increased by an amount specified by the user in the executive OPCON program. The penalized objective function is again tested for convergence. If convergence is still met, the optimizer considers the present value of the penalized objective function to be a minimum, noting again that numerical optimization programs cannot differentiate between local and global minimums.

7. The Penalty Function

The purpose of the penalty function is to increase the value of the objective function when the design is in an infeasible region. The infeasible region is that region where one or more design constraints are violated. When a constraint is violated, the value of the particular constraint,

G_j, is positive. The objective function is then penalized as follows:

$$Opj = Obj + R \cdot G_j^{\,E}$$

where:

R - a multiplication constant

E - an exponent constant

This type of penalty function, one where the penalty is applied after the design leaves the feasible region, is known as an exterior penalty function. The exterior type of penalty function was chosen over other types, such as the interior or extended interior penalty function. If a function is discontinuous within the design space being studied, numerical difficulties may be encountered which make performing an optimization of the design difficult.

III. SOLAR COLLECTOR OPTIMIZATION

A. THE CONCEPT OF NONIMAGING SOLAR COLLECTORS

At the present time there are numerous schemes for the collection of solar energy and its conversion to a more useful form of energy. In the field of solar collectors there are three broad categories: flat plate collectors, focusing collectors and nonimaging collectors. The advantages and disadvantages of each type are shown in Table I. Welford and Winston in [Ref. 3] report that "in the mid-1960's, it was realized in at least three different laboratories that light could be collected and concentrated for many purposes, including solar energy, more efficiently by nonimaging optical systems than by conventional image forming systems. The methodology of designing optimized nonimaging systems differs radically from conventional optical design. The new collectors approach very closely the maximum theoretical concentration; and for two-dimensional geometry, which is important for solar energy collection, this limit is actually reached."[1] Figure 6 shows the basic geometry for the nonimaging compound parabolic concentrating

---

[1] Welford, W. T. and Winston, R., The Optics of Nonimaging Concentrators, Light and Solar Energy, p. ix, Academic Press, 1978.

TABLE I

Collector Type Advantages and Disadvantages

COLLECTOR TYPE	ADVANTAGES	DISADVANTAGES
FLAT PLATE	Low initial cost Easy to manufacture Utilizes both direct and diffuse radiation No guidance required Good durability Low maintenance	Max. temperature 80°C Large start up losses Large convective and thermal radiation losses
FOCUSING	Max. temperature 200→3000C Low startup losses Low convective and thermal radiation losses	High manufacturing cost due to requirement for highly specular reflecting surfaces Require accurate tracking Maintenance high as reflective surface remains uncovered Does not accept diffuse radiation
NONIMAGING	Accepts both diffuse and direct radiation No tracking required Reflective surface covered	
		Moderate convective and radiation losses Moderate manufacturing costs performance improved by slight imperfections in specular reflection Moderate operating temperatures 80→300C

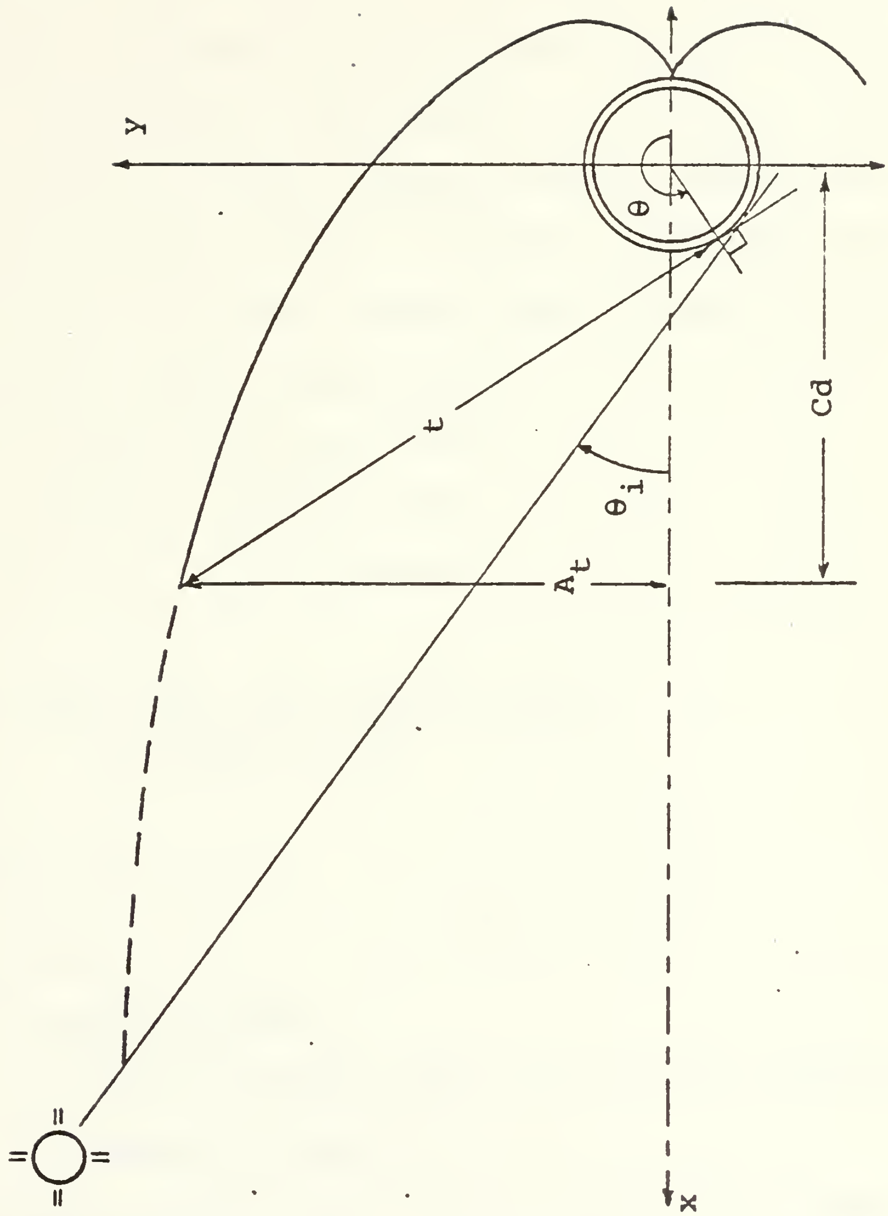


Figure 6. Nonimaging Concentrating Solar Collector Geometry

collector. Welford and Winston [Ref. 3] have shown that the concentration for a maximum input acceptance half-angle, θ_i, is obtained in two sections. The first section, that shadowed from the direct rays at angles less than θ_i is an involute of the receiver cross section. The second section is such that rays at θ_i are tangent to the receiver after one reflection at the reflector surface. The x-y coordinates of a point on the reflector surface for a collector with a circular receiver may be expressed as:

$$x = -r \cdot \cos\theta + t \cdot \cos(\theta + \pi/2)$$

$$y = r \cdot \sin\theta - t \cdot \sin(\theta + \pi/2)$$

where for $\pi/2 + \theta_i \leq \theta \leq 3\pi/2 - \theta_i$

$$t = \frac{r((\theta + \theta_i \; \pi/2) - \cos(\theta - \theta_i))}{1 + \sin(\theta - \theta_i)}$$

and for $\theta \leq \theta_i + \pi/2$

$$t = r \cdot \theta$$

r - receiver radius

θ - an angle measured from the collector centerline as shown in Figure 6.

The concentration ratio, CR, of the collector is defined as the aperature area of the collector, $2A_t$, divided by the surface area of the receiver. For the collector shown

$$CR = \frac{2A_t}{2\pi r} .$$

The collector depth, Cd, is used in the economic analysis of collector cost. The truncation angle, θ_t, is the maximum θ used in determining the collector geometry for the truncated collector. The collector may be significantly truncated before any appreciable change in the concentration ratio is affected. This allows a savings in manufacturing costs with little degradation in collector performance.

The nonimaging concentrating solar collector will accept and deliver to the absorber all incident radiation that falls on the collector aperature and that is within the maximum acceptance half angles, θ_i. That is, there is an arc of sky, $2\theta_i$, from which all radiation, direct, diffuse and reflected, is delivered to the absorber. It is this fact which makes the nonimaging concentrating collector attractive for solar energy use. Figures 7, 8, and 9 show ray paths for a nonimaging, truncated collector for the following cases: In Figure 7 the solar altitude is within the acceptance half angles. In Figure 8 the solar altitude is equal to the acceptance half angle. In Figure 9 the solar angle is less than the acceptance half angle. In Figure 7, all the radiation that enters the collector is delivered to the absorber tube and is somewhat scattered over the absorber surface. In Figure 8 again all the radiation that enters the collector is delivered to the absorber tube, but is now tangent to the tube and is concentrated on the front edge of the absorber tube. In

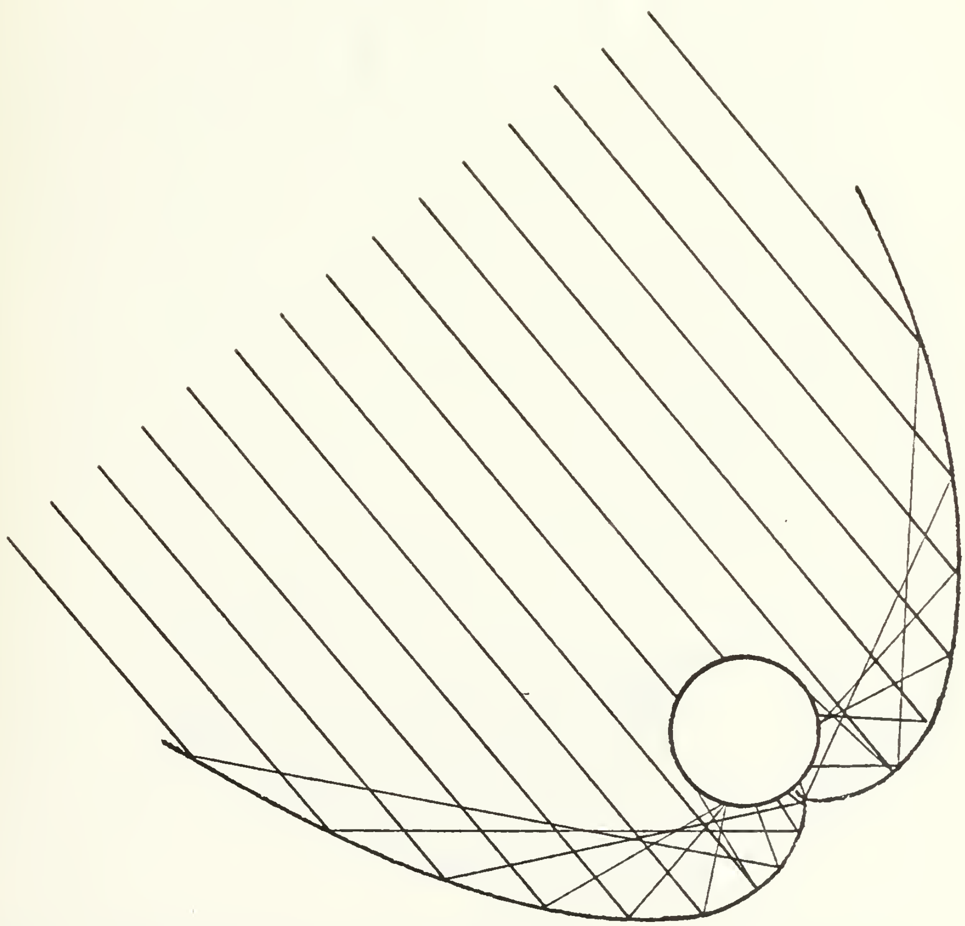


Figure 7. Ray Path Drawing for Solar Altitude Within the Maximum Acceptance Half Angle

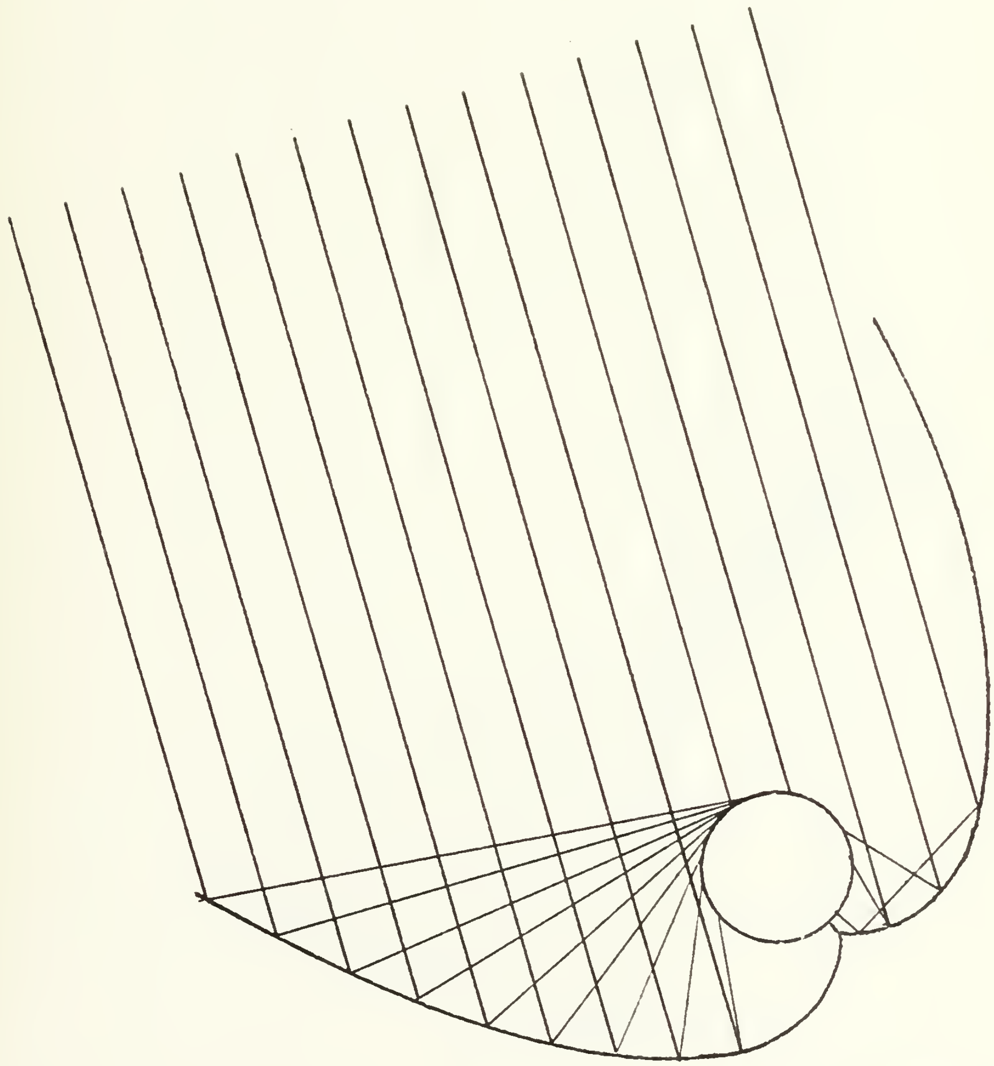


Figure 8. Ray Path Drawing for Solar Altitude Equal to the Maximum Acceptance Half Angle

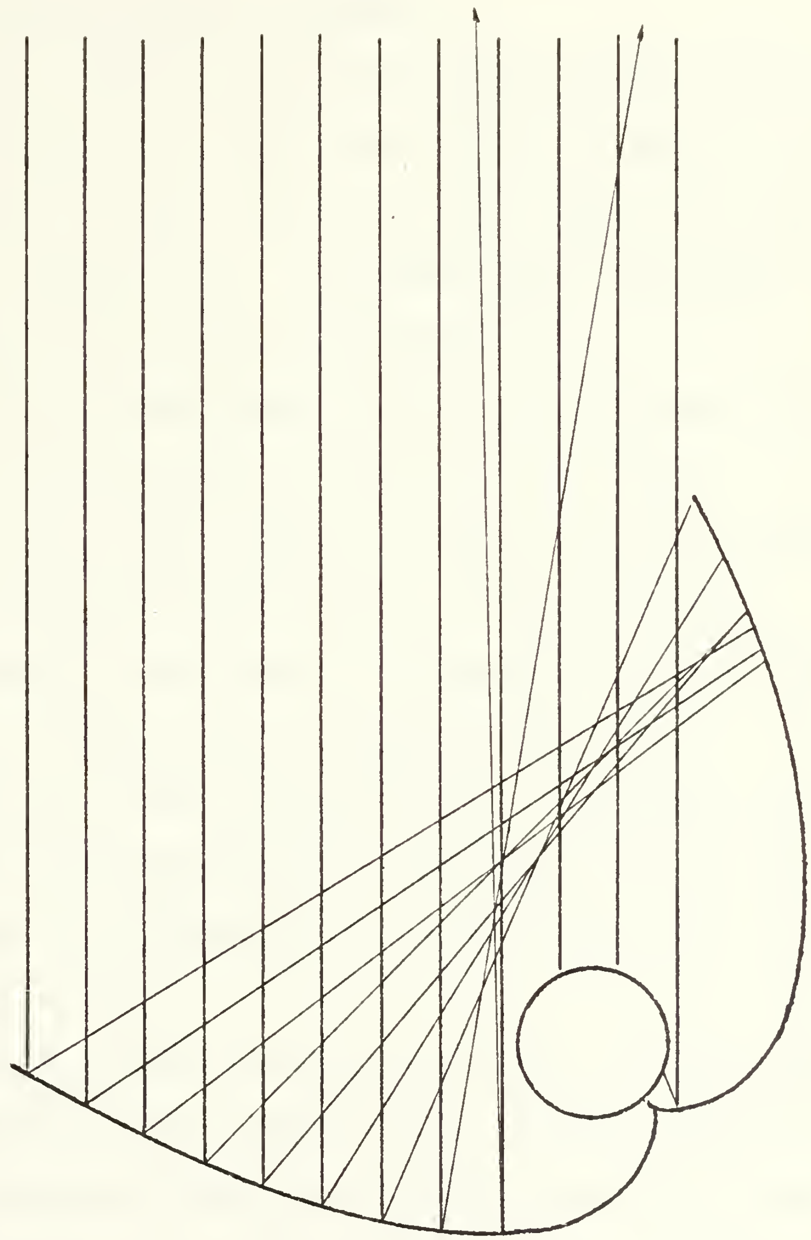


Figure 9. Ray Path Drawing for Solar Altitude Outside the Maximum Acceptance Half Angle

Figure 9 all the reflected radiation leaves the collector, thus the acceptance half angle provides a very sharp cutoff angle for accepting incident radiation.

B. SOLAR COLLECTOR DESIGN PROGRAM

A NonImaging compound parabolic trough Solar COllector hereafter referred to as NISCO was chosen to model in the analysis program. There are seven major heat flow paths considered in the program. The major heat flow paths are shown in Figure 10 and are outlined below:

q_1 - The sum of the direct, diffuse and ground reflected radiation that is incident on the collector cover.

q_2 - The sum of the direct, diffuse and ground reflected radiation that is reflected by the collector cover.

q_3 - The sum of the direct, diffuse and ground reflected radiation that is absorbed by the collector cover.

q_4 - The sum of the direct, diffuse, and ground reflected radiation that is transmitted by the cover and delivered either directly or indirectly to the absorber and is absorbed by the absorber.

q_5 - The sum of the radiation reflected by the absorber that is absorbed by the cover and the thermal radiative and convective exchanges between the absorber and the cover.

q_6 - The sum of the thermal radiative and the convective exchanges between the cover and the environment.

q_7 - The energy delivered to the collector coolant.

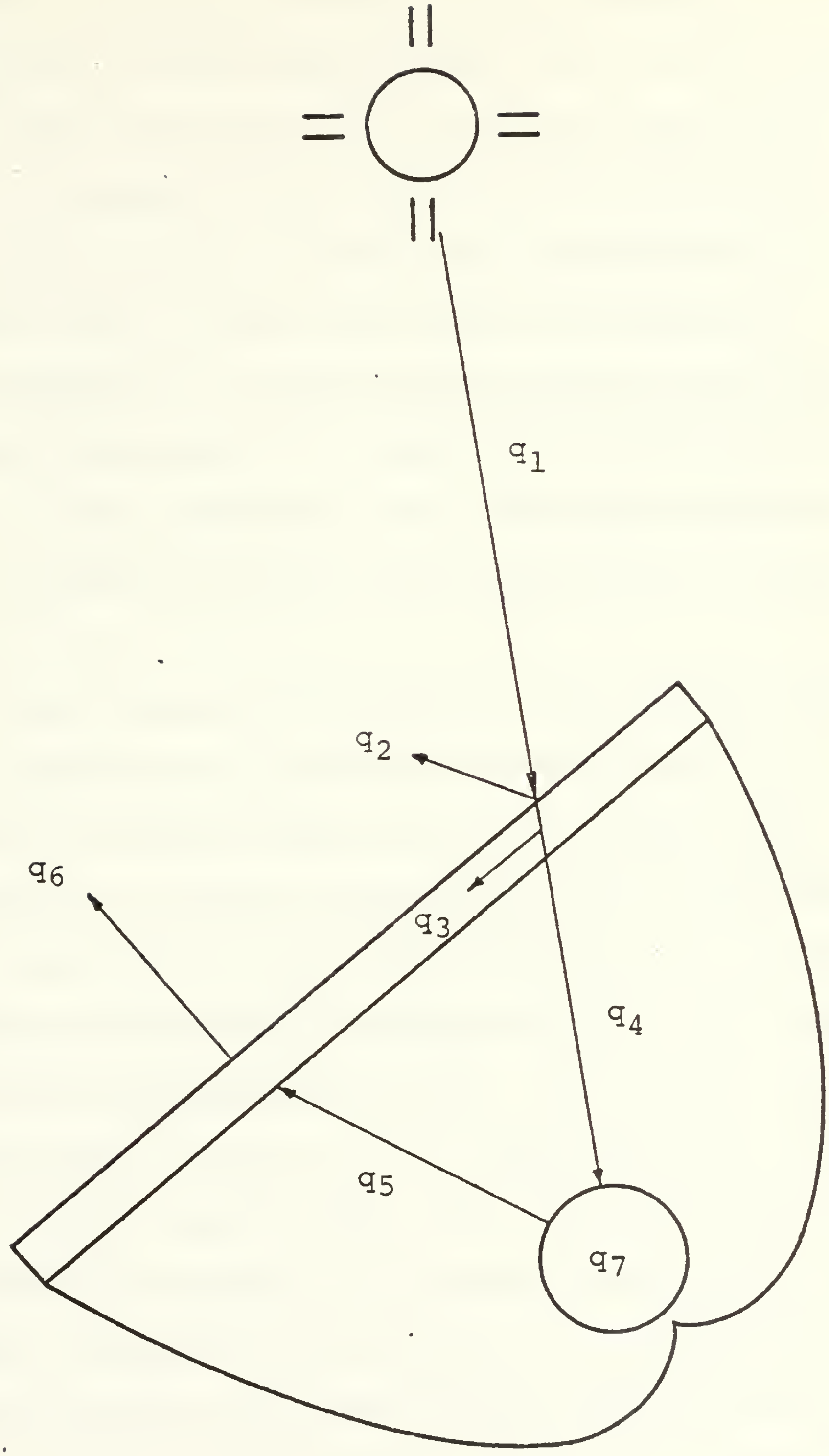


Figure 10. Basic Solar Collector Heat Paths

To determine the energy balance on the collector, a procedure outlined by Kreith and Kreider, [Ref. 6] is followed. All important heat fluxes are first calculated from basic heat-transfer principles. The fluxes are then combined in heat-balance equations for the receiver, the aperature cover, and the coolant fluid. Since the various flux terms are nonlinear in temperature a simultaneous iterative solution is used to solve the equations. Note that the first page of Appendix G is a cross reference list of the major equations used in the subprogram NISCO and the sources used to obtain the equations.

The following discussion will detail the procedure used in the NISCO subprogram to calculate the heat gain for a particular solar collector design and the resultant life-cycle fuel savings. Variable names and program line numbers correspond to those used in the NISCO subprogram. The design variables chosen for the NISCO subprogram were: (1) Thetai, (θ_i), the maximum acceptance half angle, (2) Thetat, (θ_t), the truncation angle, (3) R, (r), the receiver radius, (4) L, the collector length, and (5) Mfr, ($\dot{m}$), the coolant mass flow rate. The objective function of the NISCO subprogram is the life-cycle fuel savings of the collector. Life-cycle fuel savings are calculated as the cost savings realized by the collector over purchasing natural gas per unit quantity of 10^6 Btu's expressed in present worth using the present worth analysis described by Newnan in [Ref. 13]. The constraints

placed upon the design by the NISCO subprogram were: the maximum and minimum values allowed for the truncation angle due to collector geometry as specified in Section III.A., the mass flow rate of the coolant must be positive, the receiver radius must be positive, a minimum average daily heat gain, and a maximum allowable coolant temperature. The side constraints placed upon the design to ensure that the final design was a reasonable design were: the maximum acceptance half angle must be greater than three degrees but less than 85 degrees, the truncation angle must be greater than 185 degrees but less than 260 degrees, the receiver radius must be greater than one tenth inch but less than two inches, the collector length must be greater than five feet but less than 100 feet, and the mass coolant flow rate must be greater than five lbm/hr and less than 1000 lbm/hr.

The NISCO analysis subprogram proceeds as follows. A minimum average daily heat gain is specified and entered as Q1 in line 1655. The subprogram is then SAVED and the OPCON/DESOP optimization program run. When DESOP calls the NISCO subprogram, it will pass the design variable vector $\bar{X}$ to the NISCO subprogram. The NISCO subprogram in lines 205 to 225 sets the design variable vector $\bar{X}$ equal to the design variables used in the NISCO subprogram. NISCO then proceeds to read in the constants and data used in the solar collector design. The design specifications

are summarized in Appendix B. Collector geometry calculations are then performed in lines 820 to 855 to ascertain the optical properties of the collector as developed in [Ref. 3]. An initial receiver and aperature cover temperature is assumed in lines 865 and 870. Monthly calculations are then performed to calculate the collector tilt angle, the minimum accepted solar altitude, the ground angle factor and sky heat loss constants as specified in [Refs. 5 and 6]. An hourly calculation is then performed to calculate the angle that the sun makes with the collector aperature cover, the amount of radiation incident on the collector aperature cover and the collector cover transmissivity and absorptance as specified in [Refs. 5 and 6]. The iterative portion of the analysis then proceeds as follows: First, the heat transfer coefficients for the collector are calculated as prescribed in [Refs. 4, 6 and 11], lines 1060 to 1145. Next a heat balance is performed on the cover as prescribed in [Ref. 6] and a new collector aperature cover, Tap, is calculated in lines 1155 to 1210. A heat balance is then performed on the receiver as specified in [Ref. 6] and the energy passed to the coolant, Qu, is calculated in lines 1225 to 1305. Finally a heat balance is performed on the coolant as prescribed in [Ref. 6] and the coolant exit temperature is calculated in lines 1320 to 1365. Knowing the coolant exit temperature, Tc2, an average receiver

temperature may be calculated and compared to the initial assumed receiver temperature, lines 1375 to 1405. If the new receiver temperature is within a tenth of one percent of the old receiver temperature, the analysis program proceeds to calculate the pumping power required for the collector. If the new receiver temperature is not within the convergence specified, the iterative heat balance process is repeated, setting the aperature cover temperature and the receiver temperature to the new value calculated. The pressure drop through the collector and the power required to pump the coolant through the collector are calculated in lines 1430 to 1480, as specified in [Refs. 6 and 12]. The energy required to pump the coolant through the collector is then subtracted from the energy gained by the collector to calculate the available collector energy, Qa, line 1490. Qa is then summed for each hour that the analysis is performed. As the analysis is performed for one day each month, the summation of the available collector energy is then multiplied by thirty to obtain a yearly heat gain, Qy, line 1590. The cost of an equivalent amount of natural gas is then calculated. Using present worth analysis as described in [Ref. 13], a life-cycle savings is then calculated and the initial manufacturing cost of the collector is calculated and subtracted from the life-cycle savings. This result is then divided by the life-cycle

useful energy gain by the collector and multiplied by 10^6 to obtain the life-cycle fuel savings used as the objective function for the optimization program. The constraints on the design are then calculated in lines 1660 to 1685. Lines 1710 to 1815 contain printout specifications for the first and the last time that the NISCO subprogram is called by the optimization program. The values of the objective function and the constraints are then passed along with program control back to the optimization program.

IV. RESULTS

A major portion of the thesis work was devoted to the development of a reliable optimization program that would optimize a wide variety of problems. Following the accomplishment of this goal, a subprogram was developed to model a nonimaging concentrating compound parabolic trough solar collector.

A. RESULTS OF THE DESOP PROGRAM DEVELOPMENT

In developing the DESOP numerical optimization program, four standard numerical optimization test problems were used. The four test problems provided a wide variety of different numerical problems with which the optimization program had to deal. The four test problems are listed in Appendix F. A major goal of this thesis was to optimize all four problems using default optimizer control variables. The results of the DESOP program optimizing the four test problems are given in Appendix A. The DESOP program was able to make significant design improvements in all four test problems using default optimizer control parameters. When the optimizer control parameters were adjusted for each individual problem, the DESOP program's performance was improved in all four cases. Further experimenting with the optimizer control parameters could lead to an even better performance of the DESOP program.

B. RESULTS OF THE NISCO SUBPROGRAM

A listing of the NISCO subprogram is included as Appendix G. There are numerous comment statements in the subprogram. The reader is encouraged to look closely at the subprogram. The results of the NISCO subprogram may be found in Appendix B. The first section of Appendix B details the design specifications that were chosen for the solar collector model. Three different daily heat capacities were specified 10000, 30000 and 50000 Btu, and the NISCO subprogram was used to find the optimum design for each. The second section of Appendix B gives the initial design and final designs for the three solar collector capacities. In each case the DESOP program was able to significantly improve the design. The instantaneous efficiencies for the final designs are within a few percent of the instantaneous efficiency reported by Kreith and Kreider [Ref. 6] for a slightly different nonimaging concentrating compound parabolic trough solar collector operating under slightly different atmospheric conditions. It is interesting to note that the larger capacity solar collector had the best instantaneous efficiency and also the highest life-cycle fuel savings. In all three cases the optimum incident acceptance angle was found to be 18.04 - 18.05 degrees and the optimum truncation angle was found to lie within the range of 184.55 to 190.0 degrees.

Due to the fact that the objective function is weakly linked to the design capacity of the solar collector, the final daily heat gain is somewhat higher than the minimum set. If a stronger link were to be established between the solar collector capacity and the objective function, the collector design would be driven closer to the stated minimum daily average heat gain. Also, if the convergence criteria is tightened, the optimizer will take longer to reach the minimum but will reach a final design where the average daily heat gain is closer to the minimum set.

APPENDIX A

DESOP TEST PROGRAM RESULTS

This appendix contains the results of the four test programs that were used to develop the DESOP numerical optimization program. For each design the initial design, the true optimum design, the DESOP results using default control parameters, and the DESOP results using adjusted control parameters are given. The four test programs may be found in Appendix F.

DESOP TEST PROGRAM

ANALIZ Subprogram : BANNA

Initial Design:	True Optimum:
Design Variables:	Design Variables:
X(1) = -1.2	X(1) = 1.00
X(2) = 1.0	X(2) = 1.00
Objective Function:	Objective Function:
Obj = 10.8	Obj = 4.00
Opj = 10.8	Opj = 4.00
Side Constraints Violated:	Side Constraints Violated:
N/A	N/A
Constraints Violated:	Constraints Violated:
N/A	N/A

DESOP Results:

Default Control Parameters:	Adjusted Control Parameters
Design Variables:	Design Variables:
X(1) = Q.768	X(1) = 0.791
X(2) = 0.578	X(2) = 0.606
Objective Function:	Objective Function:
Obj = 4.05	Obj = 4.047
Opj = 4.05	Opj = 4.047
Side Constraints Violated:	Side Constraints Violated:
N/A	N/A
Constraints Violated:	Constraints Violated:
N/A	N/A
# of Function Evaluations:	# of Function Evaluations:
96	91

NOTE: The BANNA subprogram has 2 design variables and no constraints.

DESOP TEST PROGRAM

ANALIZ Subprogram : Rosen-Suzuki

Initial Design:
- Design Variables:
  - X(1) = 1
  - X(2) = 1
  - X(3) = 1
  - X(4) = 1
- Objective Function:
  - Obj = 31
  - Opj = 31
- Side Constraints Violated:
  - N/A
- Constraints Violated:
  - None

True Optimum:
- Design Variables:
  - X(1) = 0.0
  - X(2) = 1.0
  - X(3) = 2.0
  - X(4) = -1.0
- Objective Function:
  - Obj = 6.00
  - Opj = 6.00
- Side Constraints Violated:
  - N/A
- Constraints Violated
  - None

DESOP Results:

Default Control Parameters
- Design Variables:
  - X(1) = 4.72E-02
  - X(2) = 0.998
  - X(3) = 1.98
  - X(4) = -1.00
- Objective Function:
  - Obj = 6.088
  - Opj = 6.093
- Side Constraints Violated:
  - N/A
- Constraints Violated:
  - G(3) = 0.000527
- # of Function Evaluations:
  - 392

Adjusted Control Parameters
- Design Variables:
  - X(1) = -5.167E-03
  - X(2) = 1.019
  - X(3) = 1.999
  - X(4) = -0.9951
- Objective Function:
  - Obj = 5.9998
  - Opj = 6.007
- Side Constraints Violated:
  - N/A
- Constraints Violated:
  - G(3)= 0.00232
- # of Function Evaluations:
  - 306

NOTE: The Rosen-Suzuki subprogram has four design variables and three constraints.

DESOP TEST PROGRAM

ANALIZ Subprogram : T5VAR

Initial Design:
- Design Variables:
  - X(1) = 25.2
  - X(2) = 2.0
  - X(3) = 37.5
  - X(4) = 9.25
  - X(5) = 6.8
- Objective Function:
  - Obj = 3.52E-08
  - Opj = 1.64E+14
- Side Constraints Violated:
  - 1
- Constraints Violated:
  - 3

True Optimum
- Design Variables:
  - X(1) = 4.538
  - X(2) = 2.400
  - X(3) = 60.00
  - X(4) = 9.300
  - X(5) = 7.000
- Objective Function:
  - Obj = -5.26E+06
  - Opj = -5.28E+06
- Side Constraints Violated:
  - None
- Constraints Violated:
  - None

DESOP Results:

Default Control Parameters:
- Design Variables:
  - X(1) = 1.12
  - X(2) = -0.163
  - X(3) = 37.5
  - X(4) = 13.5
  - X(5) = 10.6
- Objective Function:
  - Obj = -7.36E+06
  - Opj = -7.36E+06
- Side Constraints Violated:
  - $X(2)_l$ = 1.2
  - $X(4)_u$ = 9.3
  - $X(5)_u$ = 7.0
- Constraints Violated:
  - None
- # of Function Evaluations:
  - 161

Adjusted Control Parameters:
- Design Variables:
  - X(1) = 1.72
  - X(2) = -1.56E-02
  - X(3) = 37.5
  - X(4) = 12.3
  - X(5) = 9.10
- Objective Function:
  - Obj = -6.66E+06
  - Opj = -6.65E+06
- Side Constraints Violated:
  - $X(2)_l$ = 1.2
  - $X(4)_u$ = 9.3
  - $X(5)_u$ = 7.0
- Constraints Violated:
  - G(6) = 138
- # of Function Evaluations:
  - 69

NOTE: The T5VAR subprogram has five design variables and six constraints.

DESOP TEST PROGRAM

ANALIZ Subprogram : T7VAR

Initial Design:
Design Variables:
X(1) = 1
X(2) = 1
X(3) = 1
X(4) = 1
X(5) = 1
X(6) = 1
X( ) = 1
Side Constraints Violated:
None
Objective Function:
Obj = -203
Opj = 1007
Constraints Violated:
G(1) = 11

True Optimum:
Design Variables:
X(1) = 3
X(2) = 0
X(3) = 0
X(4) = 1
X(5) = 0
X(6) = 0
X(7) = 0
Side Constraints Violated:
None
Objective Function
Obj = -190
Opj = -190
Constraints Violated:
None

DESOP Results:

Default Control Parameters:
Design Variables
X(1) = 1.42
X(2) = 0.515
X(3) = 0.376
X(4) = 0.844
X(5) = 2.13E-02
X(6) = 9.84E-03
X(7) = 0.638
Side Constraints Violated:
None
Objective Function:
Obj = -142
Opj = -142
Constraints Violated:
None
# of Function Evaluations:
302

Adjusted Control Parameters:
Design Variables:
X(1) = 2.69
X(2) = -7.60E-04
X(3) = 1.04E-02
X(4) = 1.82
X(5) = 2.89E-03
X(6) = 2.91E-03
X(7) = 0.113
Side Constraints Violated:
1
Objective Function:
Obj = -178.6
Opj = -179.1
Constraints Violated:
None
# of Function Evaluations:
808

NOTE: The T7VAR subprogram has seven design variables and one constraint.

APPENDIX B

NISCO DESIGN RESULTS

DESIGN SPECIFICATIONS

A. GEOGRAPHIC

Location	40 degrees North Lattitude
Solar position and intensity levels	ASHRAE Handbook of Fundamentals standard solar radiation design, Ref.
Wall azimuth angle	0 deg.
Cloud Cover	0 %
Vapor Pressure	3 mm Hg
Tamb	ASHRAE Handbook of Fundamentals daily norms for San Francisco, Ca. Ref.

B. COLLECTOR MATERIALS

RECEIVER	Oxidized Copper
Absoptivity	0.93
Thermal Emissivity	0.40
Reflectivity	0.07
REFLECTOR	Vacuum deposited Aluminum on resin
Reflectivity	0.89
COVER	Double strength window glass
Specular Absorptance Specular Transmisivity	ASHRAE standards for D.S. window glass. Ref.
Absorptance av.	0.03
Transmittance av.	0.80
Reflectivity av.	0.17
Thermal Emissivity	0.94
COOLANT	Therminol 55
Specific gravity	0.87
Specific heat†	4.9E-4*Tr+0.4036 Btu/(lbm F)
Viscosity†	6.71955E-4*(-0.053*Tr+32.3) lbm/(ft sec)
Inlet Temperature	100 deg. F

† Values given as a function of average receiver temperature, Tr, in degrees F.

C. ECONOMIC

Life-Cycle	20 years
Fuel Cost (Natural Gas)	8.14 $\$/10^6$ Btu
Annual Fuel Inflation Rate	0.11
Monetary Inflation Rate	0.10
Collector Cost	22.5 $\$/ft^3$ Ref. 4

NISCO DESIGN - 10000 Btu/DAY SOLAR COLLECTOR

INITIAL DESIGN

Design Variables:		
Incident acceptance half angle	25.00	deg
Truncation angle	230.0	deg
Receiver radius	0.41	in
Collector length	20.00	ft
Coolant mass flow rate	108.7	lbm/hr
Design Features:		
Concentration ratio	2.20	
Collector aperature area	9.56	ft^2
Collector depth	4.12	in
Coolant velocity	0.15	ft/sec
Average daily heat gain	11,400	Btu
Maximum coolant temperature	137	deg F
Instantaneous collector efficiency	0.70	
Initial cost	\$203.00	
Life-cycle cost savings	\$7.78	$/10^6$ Btu

FINAL DESIGN

Design Variables:		
Incident acceptance half angle	18.05	deg
Truncation angle	190.0	deg
Receiver radius	0.52	in
Collector length	18.92	ft
Coolant mass flow rate	124.3	lbm/hr
Design Features:		
Concentration ratio	1.65	
Collector aperature area	8.48	ft^2
Collector depth	1.02	in
Coolant velocity	0.11	ft/sec
Average daily heat gain	10,600	Btu
Maximum coolant temperature	131	deg F
Instantaneous collector efficiency	0.74	
Initial cost	\$159.00	
Life-cycle cost savings	\$8.15	$/10^6$ Btu

NISCO DESIGN - 30000 Btu/DAY SOLAR COLLECTOR

INITIAL DESIGN:

Design variables:		
Incident acceptance half angle	15.00	deg.
Truncation angle	230.0	deg.
Receiver radius	1.00	in.
Collector length	50.0	ft.
Coolant mass flow rate	100.0	lbm/hr
Design features:		
Concentration ratio	2.95	
Collector aperature area	77.35	ft^2
Collector depth	12.62	in.
Max. coolant temperature	366.59	F
Instantaneous collector efficiency	0.64	
Average daily heat gain	83,400	Btu
Initial cost	$1,628.90	
Life cycle savings	$7.54	$/10^6$ Btu

FINAL DESIGN:

Design variables:		
Incident acceptance half angle	18.04	deg.
Truncation angle	184.55	deg.
Receiver radius	1.28	in.
Collector length	26.43	ft.
Coolant mass flow rate	425.23	lbm/hr
Design features:		
Concentration ratio	1.54	
Collector aperature area	27.35	ft^2
Collector depth	1.78	in.
Coolant velocity	6.04E-02	ft/sec
Max. coolant temperature	129.4	F
Instantaneous collector efficiency	0.75	
Average daily heat gain	34,473	Btu
Initial cost	$509.98	
Life-cycle savings	$8.19	$/10^6$ Btu

Run time - approx. 5 hours

NISCO DESIGN - 50000 Btu/DAY SOLAR COLLECTOR

INITIAL DESIGN

Design Variables:		
Incident acceptance half angle	20.00	deg.
Truncation angle	220.0	deg.
Receiver radius	1.25	in.
Collector length	50.00	ft.
Coolant Mass flow rate	500.0	lbm/hr
Design Features:		
Concentration ratio	2.28	
Collector aperature area	75.5	ft^2
Collector depth	9.14	in.
Coolant velocity	7.51E-02	ft/sec
Maximum coolant temperature	164.	deg. F
Instantaneous efficiency	0.72	
Average daily heat gain	91,900	Btu
Initial cost	$1,510.00	
Life-cycle savings	$7.96	$/10^6$ Btu

FINAL DESIGN

Design Variables:		
Incident acceptance half angle	18.04	deg.
Truncation angle	185.0	deg
Receiver radius	1.43	in
Collector length	39.3	ft
Coolant mass flow rate	986	lbm/hr
Design Features:		
Con entration ratio	1.55	
Collector aperature area	45.5	ft^2
Collector depth	2.04	in
Coolant velocity	0.11	ft/sec
Maximum coolant temperature	121	deg F
Instantaneous collector efficiency	0.76	
Average daily heat gain	57,800	Btu
Initial cost	$848.00	
Life-cycle savings	$8.21	$/10^6$ Btu

APPENDIX C

DESOP USER'S MANUAL

A. INTRODUCTION

The numerical optimization package of programs, currently consisting of OPCON and DESOP, provide the capability for finding the optimum design of a system mathematically modeled using multiple variables, on the Hewlett-Packard 9845A desk-top computer. OPCON is an executive program which provides the user with the following: a primary point of contact with the computer from which to access the optimization program DESOP, a standard formatted input for design variables, side constraints on the design variables, optimizer control variables, and organization of the optimization process. DESOP is a numerical optimization program, which when coupled to a user supplied design analysis program, will optimize the design. DESOP will allow the user to monitor the optimization process as it is taking place. After monitoring the optimization process, the user may choose to change the optimizer control variables and/or his design starting point to more efficiently or more accurately reach the design optimum. After an optimization has been performed, DESOP will reload the OPCON program and return the user to the OPCON program. The OPCON program will then offer the

user the following three choices: to optimize the same design, optimize a different design or terminate the program.

Figure 11 illustrates the overlay method used in loading the OPCON and DESOP programs.

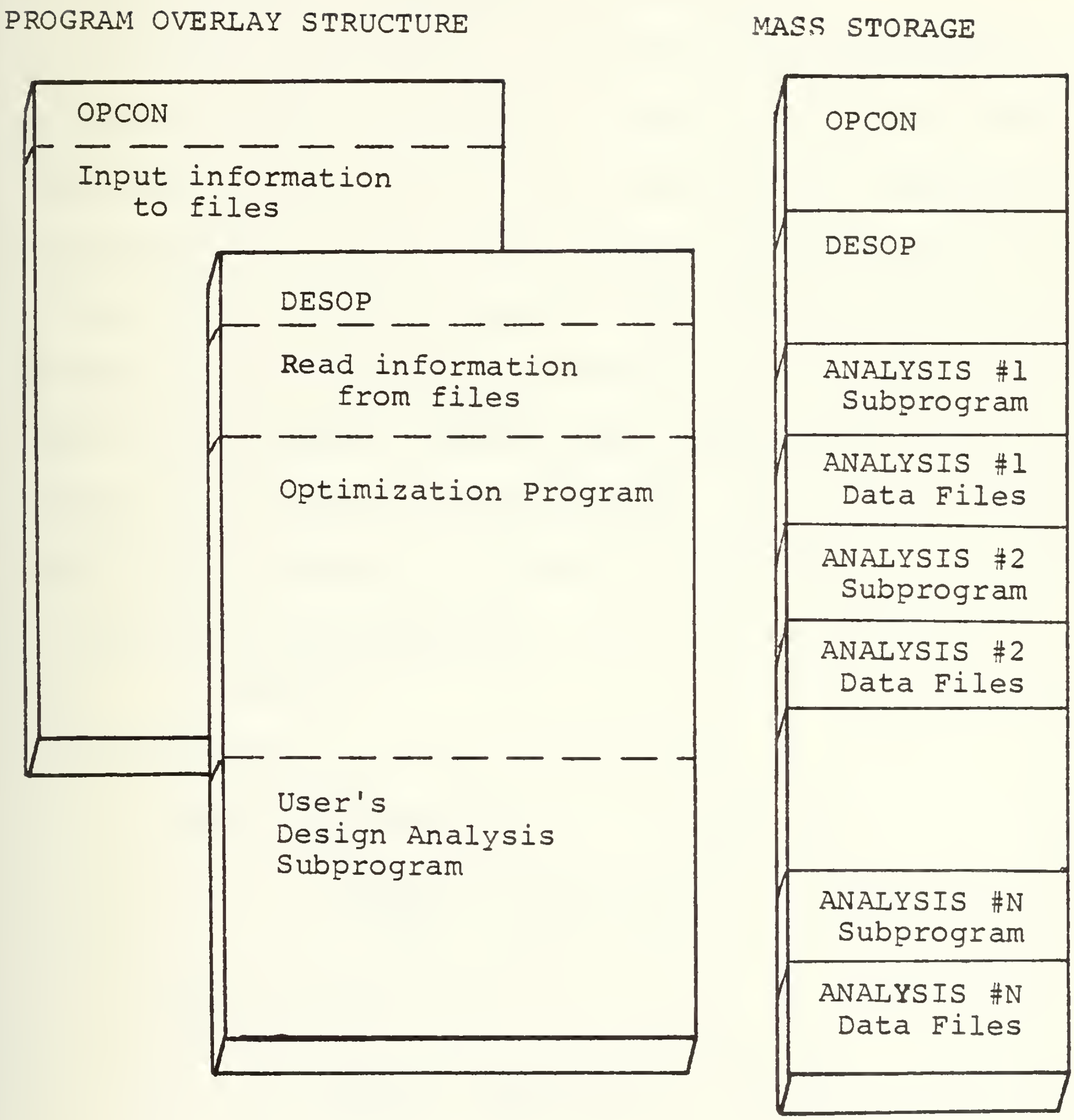


Figure 11. Basic Program Relationships

B. THE CONCEPT OF NUMERICAL OPTIMIZATION

Consider the problem of designing the cantilevered beam shown in Figure 1. The design task may be broken down into three major parts. First, the objective of the design must be determined, which in this case is to minimize the weight of the beam required to support the concentrated tip load P. Second, any physical constraints that may effect the design must be determined. Thirdly, any limits which exist on the design variables must be determined. The design problem may then be reduced to a system of equations as follows:

Minimize the volume (V)

$$V = B \cdot H \cdot L$$

Subject to:

Bending stress (σ_b)

$$\sigma_b = \frac{6 \cdot P \cdot L}{B \cdot H^2} \le 20000 \text{ psi}$$

Shear stress (ν)

$$\nu = \frac{3 \cdot P}{2 \cdot B \cdot H} \le 10000 \text{ psi}$$

Deflection under load (δ)

$$\delta = \frac{4 \cdot P \cdot L^3}{E \cdot B \cdot H^3} \le 1 \text{ inch}$$

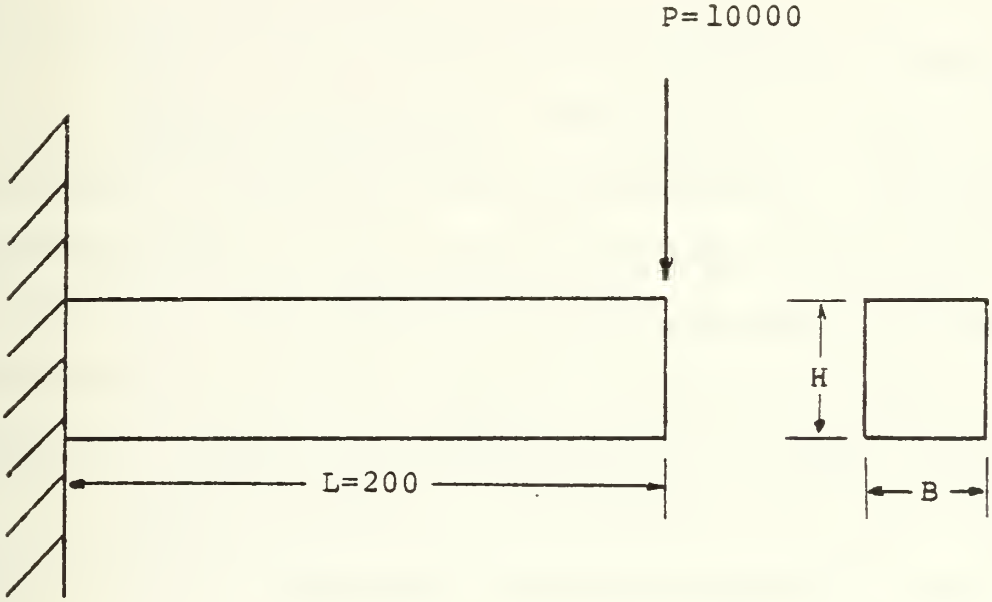


Figure 1. Cantilevered Beam Design Problem

With geometric constraints such that:

$$0.5 \leq B \leq 0.5$$

$$1.0 \leq H \leq 20.0$$

$$H/B \leq 10.0$$

At this time the following definitions are introduced:

Objective Function - The parameter which is to be minimized or maximized during optimization. The objective function always occurs on the left side of the equation unless it is also used as a design variable. An objective function may be either linear or nonlinear, implicit or explicit, but must be a function of the design variables.

Design Variable - Any parameter which the optimization process is allowed to change in order to improve the design. Design variables appear only on the right hand side of equations in the analysis program.

Inequality Constraint - Any parameter which must not exceed specified bounds for the design to be acceptable. Constraint functions always appear on the left side of equations. A constraint may be linear, nonlinear, implicit or explicit, but must be a function of one or more design variables.

Equality Constraint - Any parameter which must equal a specified value for the design to be acceptable. The same rules apply to equality constraints as inequality constraints.

Side Constraint - Any upper or lower bound placed upon a design variable. Side constraints are usually not included in the system of equations that comprise an analysis program. Instead they are usually included as part of the data input to the optimization program.

Analysis Code - The system of equations utilizing the design variables which are used to calculate the objective function and the constraints of a particular design problem.

The general optimization problem may thus be stated mathematically as:

Find the set of design variables $\bar{X}_i$ where $i = 1,2,...,n$ which will:

Minimize the objective function (Obj)

$$Obj = f(\bar{X})$$

Subject to:

Inequality constraints (G)

$$G_j(\bar{X}) \leq 0 \qquad j = 1,2,...,m$$

Equality constraints (H)

$$H_j(\bar{X}) = 0 \qquad j = 1,2,...,l$$

Side constraints

$$X_i^l \leq X_i \leq X_i^u \qquad i = 1,2,...,n$$

Returning to the cantilevered beam problem, it may be stated in the standard format as follows:

Let $X(1) = B$, $X(2) = H$, and $Obj = Vol = B \cdot H \cdot L$

Then minimize $Obj = Vol$

Subject to:

$$G(1) = \frac{\sigma_b}{20000} - 1 \le 0$$

$$G(2) = \frac{\nu}{10000} - 1 \le 0$$

$$G(3) = \delta - 1 \le 0$$

$$G(4) = \frac{H}{B} - 10 \le 0$$

With side constraints:

$$X(1)^l = 0.5$$

$$X(1)^u = 5.0$$

$$X(2)^l = 1.0$$

$$X(2)^u = 20.0$$

It is thus fairly simple and straightforward to perform an analysis on a particular beam for a given B and H. Successive analyses may be performed on the cantilevered beam by solving the above system of equations. It is desirable to automate the successive solutions and to direct the solutions such that each solution is a better design than the last. One approach for doing so, and the one used by DESOP is to proceed as follows: Start with initial values for B and H. Solve the above set of equations to find the objective function Obj and to see if any constraints are violated. A pseudo objective function is created to represent designs when constraints are violated. If a constraint is violated, a penalty is added to Obj to form a penalized objective function Opj. The gradient of the

penalized objective function at the initial design may be found by taking the first partial derivative of Opj with respect to the design variables. The gradient of the penalized objective function defines the direction of steepest ascent. In the case of the cantilevered beam, it is desired to minimize the objective function; therefore, the greatest improvement in design may be achieved by moving in the negative gradient, or steepest descent direction. From the initial design point a search is performed in the steepest descent direction for the minimum value of Opj in that direction. At the new minimum, the gradient of the penalized objective function is again determined and a search is performed in a conjugate direction until a second minimum is found. Successive iterations are performed until the gradient is found to be zero or each successive iteration produces a sufficiently small change in Opj such that for all practical purposes the minimum has been found. At this time the penalty function is increased. If the design is in a region where there are no constraints violated an increase in the penalty function will not change the value of Opj. If on the other hand the design is in an infeasible region where there are one or more constraints violated, Opj will be increased, and the search for a new minimum will commence. If the minimum of the objective function exists in the infeasible region,

the minimum value for the objective function in the feasible region will be approached from the infeasible region as the penalty function is increased. The design improvement process will terminate when a zero gradient is found or successive iterations produce a sufficiently small change in the value of Opj and an increase in the penalty function causes no change in Opj.

The minimum thus reached by the optimization process is a minimum with respect to the penalized objective surface immediately surrounding the final design point. The optimization process cannot distinguish between local and global minimum points. It is thus good engineering practice to run several optimizations for a particular design problem from several different initial design points. If optimizations performed from different initial design points converge on the same minimum point, that point is probably a global minimum. If on the other hand two or more minimums are found, there may be local minimums located in the design space being considered and care must be taken to find the global minimum.

C. THE DESOP NUMERICAL OPTIMIZATION PROGRAM

The DEsktop Sequential unconstrained minimization technique Optimization Program was developed using the basic optimization approach outlined in Section II.A. A copy of the program is included in Appendix E. The major

program structure is shown in Figure 2. The following discussion will refer to Figure 2 and describe the major features of the program.

1. Basic Program Execution

The DESOP program begins execution when it is loaded, linked to the user's analysis subprogram, and the program is instructed to run by the OPCON program. The above actions are automatically performed by the OPCON program. DESOP is loaded into the computer by an overlay process. Therefore no variables can be directly transferred between the DESOP and OPCON programs. DESOP begins execution by reading the optimizer control variables and the design variables that were input using the OPCON program and saved to a mass storage device. The program then sets Icalc equal to one and evaluates the objective function and constraints at the initial design point. Icalc is a flag provided the user to key user specified output on the initial and final design analysis. DESOP will provide the user with a hard copy output of the initial design variables, the value of the constraints, the objective function and the penalized objective function. The user then has the option of continuing with the DESOP program to optimize his analysis subprogram or to return to the OPCON program to change one or more of the input parameters.

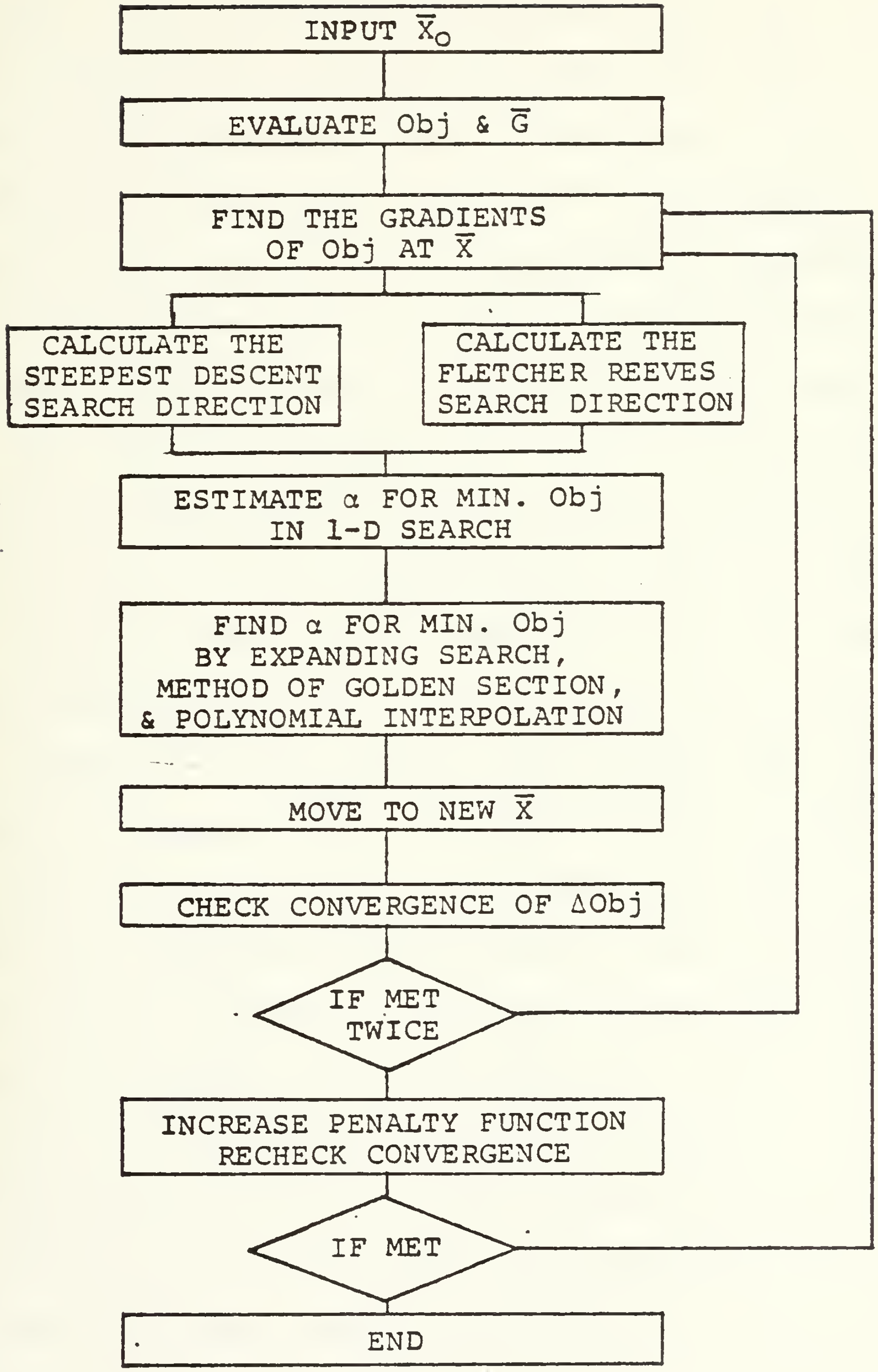
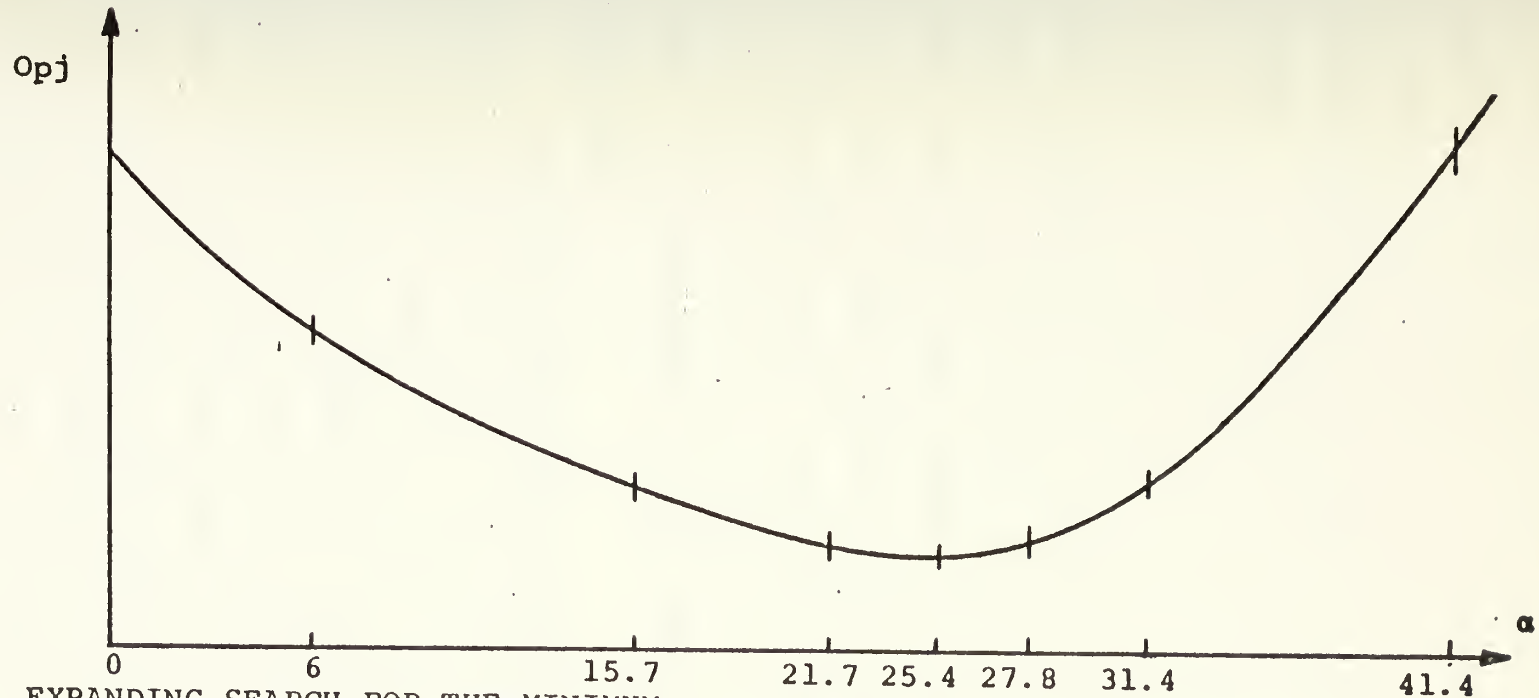


Figure 2. DESOP Flow Diagram

Proceeding with the optimization, there are two major loops in the optimization program. The outer loop increases the penalty function when the inner loop's convergence criteria have been met. A convergence test is then performed by the outer loop. If the convergence criteria is met, the optimization process is considered finished. If the convergence criteria for the outer loop is not met, program execution is returned to the inner loop. The inner loop performs successive iterations searching for the minimum of the penalized objective function with no increase in the penalty function taking place. When the inner loop's convergence criteria have been met program execution is transferred to the outer loop.

Execution of the program while in the inner loop proceeds as follows: First, the gradient of the penalized objective function is calculated by subroutine GRAD. The program then computes a search direction using either a steepest descent method or the method of conjugate directions developed by Fletcher and Reeves [Ref. 1]. Once a search direction is established the optimizer attempts to locate the minimum value of the penalized objective function in the search direction. This process if referred to as the one-dimensional search and is illustrated in Figure 3. The efficiency and accuracy to which the one-dimensional search for the minimum of the penalized objective function



EXPANDING SEARCH FOR THE MINIMUM

αz^1	$\alpha 3^1$							
αz^2	$\alpha 1^2$	$\alpha 3^2$						
	αz^3	$\alpha 1^3$						$\alpha 3^3$

MINIMUM BRACKETED - GOLDEN SECTION SEARCH TO LOCALIZE THE MINIMUM

$Opj1^4 < Opj2^4$	αz^4		$\alpha 1^4$		$\alpha 2^4$	$\alpha 3^4$
$Opj1^5 > Opj2^5$	αz^5	$\alpha 1^5$	$\alpha 2^5$		$\alpha 3^5$	
		αz^6	$\alpha 1^6$	$\alpha 2^6$	$\alpha 3^6$	

Figure 3. The One-Dimensional Search Process

is accomplished, is the key to successful sequential unconstrained minimization technique numerical optimization. The one-dimensional problem may be expressed in terms of the penalized objective function, Opj, and the amount of movement, α, in the search direction, $\bar{S}$. First the slope of Opj with respect to α is calculated. An initial "guess" of how far to move is made using subroutine ALPGES. The α which corresponds to the minimum value of Opj in the one-dimensional search is then calculated using subroutine ALPBND and subroutine QUEBIC. The minimum value of Opj thus found is then compared to the previous value of Opj for convergence using subroutine CONVRG. If convergence is not met, execution returns to the start of the inner loop. If convergence is met, execution returns to the outer loop.

When the convergence criteria have been met for both the inner and outer loops, the program proceeds to set ICALC to three as a flag for user generated output for the final design. DESOP then provides the user with a hard copy output of the final design variables, objective function value, penalized objective value, constraint values, the number of inner loop iterations, the number of times the analysis subprogram was called and the final value of the penalty function. The OPCON program is then overlayed over the DESOP program and program execution is returned to the OPCON program.

2. Finding the Search Direction

The first step in finding the search direction, $\bar{S}$, is to determine the slope of Opj at the present design point. The forward finite difference method is used where:

$$\frac{\partial F}{\partial X_i} = \frac{F(X_i + \Delta X_i) - F(X_i)}{\Delta X_i} = -S_i$$

$$i = 1,2,...Ndv$$

As $\partial F/\partial X_i$ gives the direction of positive slope, the search direction is the negative of $\partial F/\partial X_i$. The first search is performed using the steepest descent as found above using the following relation:

$$X_i' = X_i + \alpha S_i$$

where alpha is the distance moved in the $\bar{S}$ direction. When a minimum is obtained along the direction of steepest descent, a new Fletcher-Reeves conjugate search direction [Ref. 1] is calculated at the new Design point using the following relations:

$$S_i' = \frac{\partial F}{\partial X_i} + BS_i$$

$$B = \frac{\sum_{i=1}^{Ndv} \left[\frac{\partial F'}{\partial X_i}\right]^2}{\sum_{i=1}^{Ndv} \left[\frac{\partial F}{X_i}\right]^2}$$

where the prime denotes values for the present iteration and the non-prime variables indicate values for the previous

iteration. A one-dimensional search is then performed in the new search direction. Searches are continued using the conjugate direction method for Ndv + 1 iterations, where Ndv is the number of design variables. The search process is then restarted using the steepest descent method. The reason for incorporating the conjugate search method is that the steepest descent method when traversing a design surface with a curved valley will tend to zigzag from one side of the design surface valley to the other making very little progress as is illustrated in Figure 4. The conjugate direction method is much more efficient in traversing such a design surface. However, as the conjugate direction method is additive upon previous searches, it has a tendency to decrease in effectiveness with each successive search owing to the accumulation of numerical "noise." For that reason the search process is restarted with the steepest descent method every Ndv + 1 iterations, or when the conjugate direction predicts a positive slope. The search direction is normalized to avoid inaccuracies caused by numerical ill-conditioning.

3. Estimating an Initial Value for Alpha

The initial estimate for alpha is made in the following manner: First, the slope of Opj in the search direction is calculated as the sum of each of the products of the gradients times the search direction. Then the slope of Opj in the search direction is divided by the value

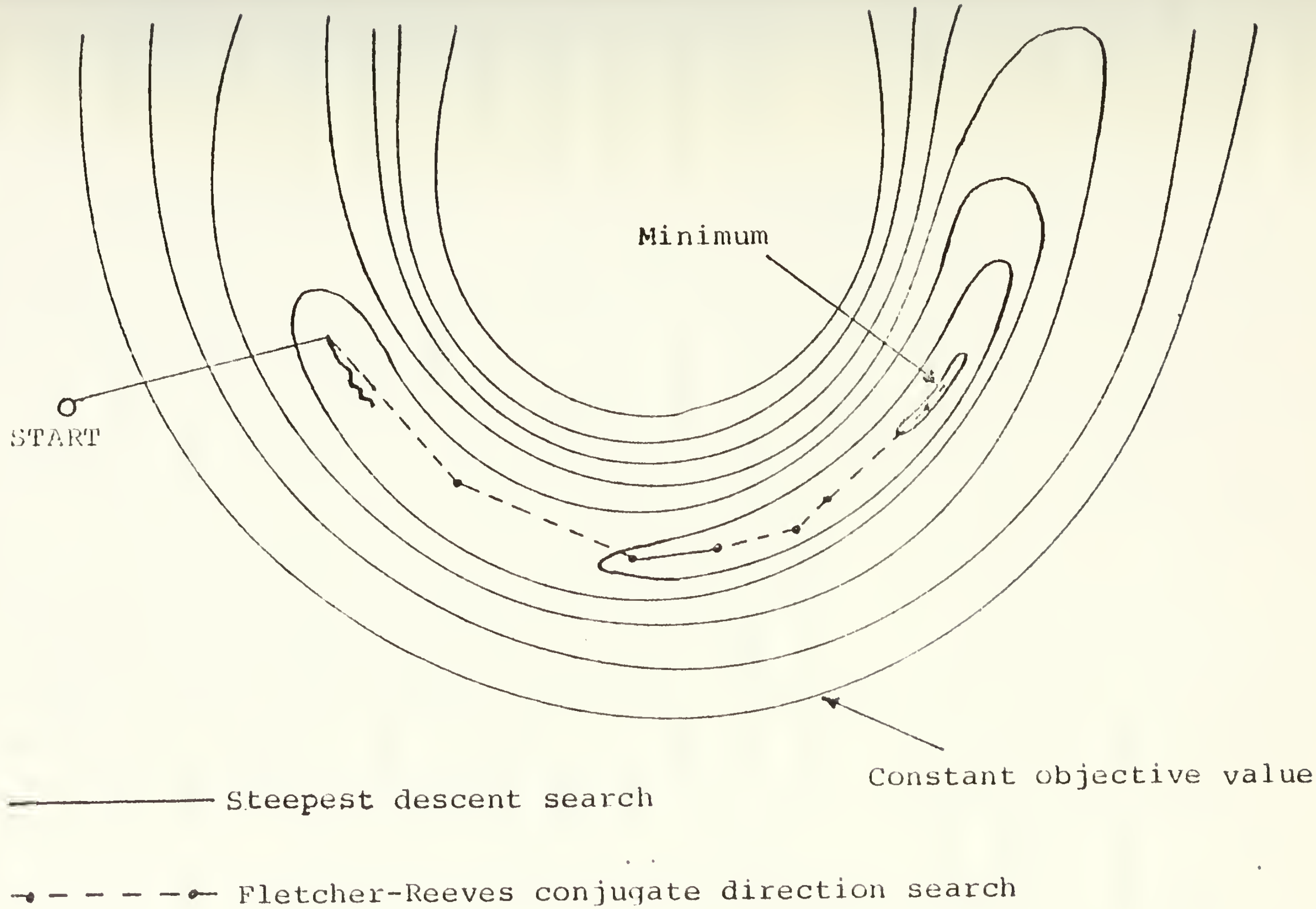


Figure 4. Steepest Descent and Conjugate Direction Search in a Two-Dimension Design Space.

of Opj. This value is then multiplied by an improvement percentage in Opj. This first estimate is then applied to a series of conditional tests to determine the validity of the estimate with respect to the slope of Opj and the magnitude of the design variables. Lastly, the estimate for alpha is checkec to see if it violates any side constraints. If it does, the value of the estimate for alpha is reduced until the side constraints are no longer violated.

4. Calculating Alpha

The calculation of alpha is the most critical algorithm in the DESOP program in providing reliable optimizer operation. The ability to accurately and efficiently find the minimum of the penalized objective function in the one-dimensional search affects directly the operation of the optimizer. Figure 5 illustrates the zigzag phenomenon which occurs if alpha is not accurately found. The zigzag phenomenon is caused by the fact that the optimizer in performing the forward finite difference for calculating the search direction perturbs the design vector a very small amount. As such the optimizer can only "see" the design surface that is immediately adjacent to the design point. Therefore, if the minimum is not found in the one-dimensional search, the optimizer will converge very slowly on the minimum.

There are tow major sections to the ALPBND subroutine. The first section attempts to find the minimum value of Opj

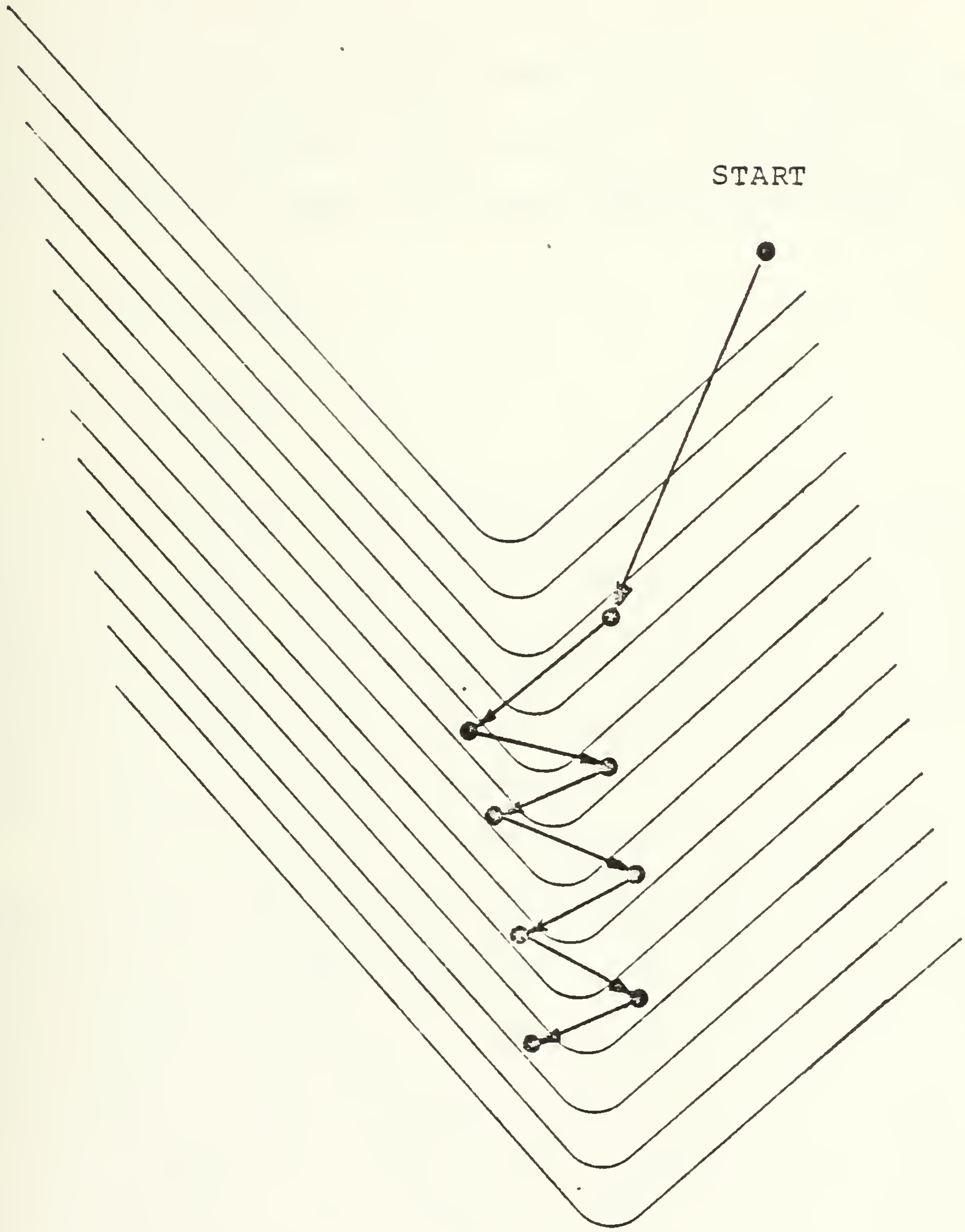


Figure 5. The Zigzag Phenomenon

using an expanding search technique. The first move is the amount predicted by the ALPGES subroutine. If the minimum is not bracketed by the first move, additional moves are made. Each move is larger than the last. The size of the move is increased each time by an amount equal to the size of the last move divided by the lower Golden Section fraction, where the Golden Section fractions are:

$$F_1 = \frac{3 - \sqrt{5}}{2}$$

$$F_2 = \frac{\sqrt{5} - 1}{2}$$

The lower Golden Section fraction, F_1, is used so that the interval will be consistent with the Golden Section search in the second section of the ALPBND subroutine. The expanding search is continued until the minimum value of the objective function has been bracketed.

Once the minimum is bracketed, a Golden Section search is performed to reduce the bracketing interval on the minimum by an amount such that when the two end points of the interval are taken with two points internal to the interval and a cubic is passed through the four points, the cubic will accurately predict the minimum of the penalized objective function. Himmelblau in [Ref. 2] states that the Golden Section search method of reducing the interval around the minimum of Obj is the most effective of the reducing techniques studied. Golden Section search is based

on the splitting of a line into two segments known in ancient times as the "Golden Section." The ratio of the whole line to the larger segment is the same as the ratio of the larger segment to the smaller segment. The two Golden Section fractions are employed to split the interval bracketing the minimum as shown in Figure 3. Once the interval has been split, the two values of Opj corresponding to the internal points are compared to find the larger of the two. The internal point with the larger value of Opj will become the new end point for the interval, the remaining interior point will by the fact that it was determined by a Golden Section fraction, be equal to the point determined by the other Golden Section fraction. Thus, only one new point must be calculated to continue the Golden Section search. The search is continued in this manner until the vertical separation of the two end points with respect to the interior points is less than one percent. The four values of the penalized objective function corresponding to the four Golden Section search points are then sent to a cubic interpolator. The cubic interpolator will return a value for alpha to predict the minimum of the penalized objective function, and the minimum of the cubic function that the interpolator has created. The subroutine ALPBND will then test the predicted minimum with the minimum found at the predicted alpha. If there is less than a tenth of

one percent difference between the two values of the objective function, the point predicted by the cubic interpolator will be accepted as the minimum and program execution will return to the main program. If the predicted minimum is not sufficiently close to the minimum at the predicted alpha, another Golden Section search will be performed to reduce the interval and better localize the minimum. The four points from the reduced interval will then be sent to the cubic interpolation subroutine. This process will continue until either the test for the minimum is positive or the interval has been reduced to less than 1E-12. Program execution will then return to the main program.

5. Subroutine QUEBIC

Subroutine QUEBIC is used to estimate the alpha at which Opj is a minimum based on four point cubic interpolation. If the function more closely resembles a quadratic than a cubic, a three point quadratic interpolation is performed using the three points which bracket the minimum. If the predicted minimum is outside the interval spanned by the two end points again a quadratic interpolation is performed. If the minimum still lies outside the two end points, the analysis returns to subroutine ALPBND, the inverval bracketing the minimum is reduced, and program execution returns to QUEBIC.

6. Convergence of the Penalized Objective Function

The penalized objective function is tested for convergence at the end of each inner loop and again at the end of the outer loop in the main program. Convergence is tested by calling subroutine CONVRG. There are two criteria used for testing for convergence. The first tests the relative difference of the value of Opj from the present iteration with the value of Opj from the last iteration. The second method tests the absolute difference of the two values. The second method is employed for cases when the value of the penalized objective function approaches zero. When convergence has been met on two successive iterations, the penalty function is increased by an amount specified by the user in the executive OPCON program. The penalized objective function is again tested for convergence. If convergence is still met, the optimizer considers the present value of the penalized objective function to be a minimum, noting again that numerical optimization programs cannot differentiate between local and global minimums.

7. The Penalty Function

The purpose of the penalty function is to increase the value of the objective function when the design is in an infeasible region. The infeasible region is that region where one or more design constraints are violated. When a constraint is violated, the value of the particular constraint,

G_j, is positive. The objective function is then penalized as follows:

$$Opj = Obj + R \cdot G_j E$$

where:

R - a multiplication constant

E - an exponent constant

This type of penalty function, one where the penalty is applied after the design leaves the feasible region, is known as an exterior penalty function. The exterior type of penalty function was chosen over other types, such as the interior or extended interior penalty function. If a function is discontinuous within the design space being studied, numerical difficulties may be encountered which make performing an optimization of the design difficult.

D. USE OF THE DESOP NUMERICAL OPTIMIZATION PROGRAM

Once the analysis subprogram has been written and SAVED, the user may perform a numerical optimization of the design analysis. Figure 11 shows the basic relationships between the OPCON and DESOP programs and the user supplied ANALIZ subprograms. Note that the DESOP program overlays the OPCON program after all data has been input. To begin, LOAD OPCON and press run. Through a series of self-explanatory menus, the user will be prompted to input control variables, design variables and execute the DESOP program. The following menus appear in the OPCON program.

1. INTRODUCTION - A brief introduction stating the purpose of the OPCON program.

2. NEW OR EXISTING ANALYSIS PROGRAM - If this is the first time that a particular analysis subprogram has been run on the tape or disk being used, or if the files from a subsequent run have been deleted, the user must enter the response for a new analysis subprogram. For existing programs, OPCON will read data from the existing data file. The data read will be that from the subsequent run of the particular analysis subprogram.

3. NAME THE ANALIZ SUBPROGRAM - The user will input the name under which the analysis subprogram he wishes to use was saved. If this is a new analysis subprogram OPCON will create files for saving the data input during the execution of OPCON.

4. OPTIMIZER SELECTION - At the present time there is only one optimization program available. It is hoped that at some future date additional numerical optimization programs will be added to the optimization package. OPCON has been developed to interface with multiple numerical optimization programs. Details for linking other numerical optimization programs to OPCON are included as comments in the OPCON listing which may be found in Appendix D.

5. INPUT NDV AND NCON - For a new analysis subprogram the user will be asked to input the number of design variables, NDV, and the number of constraints in the analysis subprogram, NCON.

6. INPUT CHECK OF COMMON CONTROL VARIABLES - A menu of the optimizer control parameters common to DESOP, Feasible Direction type and other future optimization programs is displayed. The menu displays the variable name, its minimum, maximum, default and present value. The variables displayed are:

NDV - The number of independent design variables used in the analysis code.

NCON - The number of constraints in the design analysis subprogram.

IPRINT - A print control used in the optimization program to display intermediate results. Positive values entered will print on the CRT. Negative values entered

will print on the thermal printer. If a zero is entered there will not be a hard copy output of the initial and final results of the optimization program. IPRINT = 0 is to be used when debugging an analysis subprogram to conserve thermal paper.

± 1 - Print initial and final optimization information.

± 2 - Print above plus the objective function and penalized objective function on each iteration.

± 3 - Print above plus the constraint values, search direction vector and move parameter alpha on each iteration.

± 4 - Print above plus gradient information on each iteration.

± 5 - Print above plus each proposed design vector and the penalized objective function during the one-dimensional search on each iteration.

± 6 - Debugging aid for optimizer development. DESOP will pause after each major operation is performed during the optimization process.

DELFUN - The minimum absolute change in the objective function to indicate convergence of the optimization process.

DABFUN - The minimum absolute change in the objective function to indicate convergence of the optimization process.

ITMAX - The maximum number of inner loop (unconstrained minimizations) without increasing the penalty function.

ICNDIR - The conjugate direction restart parameter. Every ICNDIR inner loop iterations a steepest descent search is performed. It is recommended that ICNDIR be set equal to NDV + 1.

FDCH - The relative change in the design variables for calculating finite difference gradients.

FDCHM - The minimum absolute step in finite difference gradient calculations.

ABOBJ1 - The expected fractional change in the objective function for the first estimates of the step size to be taken in the one-dimensional search.

ALPHAX - The maximum fractional change in any design variable for the first estimate of the step size to be taken in the one-dimensional search.

7. INPUT CHECK OF THE DESOP CONTROL PARAMETERS - A menu of the optimizer control parameters for the DESOP optimization program is displayed. The menu will display the variable name, its maximum, minimum, default and present value. The variables displayed are:

IRMAX - The maximum number of times that the penalty parameter will be increased.

RZ - The starting value of the penalty parameter.

RMULT - The amount by which RZ is multiplied each time that it is increased.

EXPG - The amount by which a violated constraint value is raised to an exponent, EXPG.

NSCAL - A design variable auto-scaling control.

0 - No scaling of the design variables is performed by DESOP.

1 - The design variables are scaled every ICNDIR iteration.

9. INPUT DESIGN VARIABLES AND SIDE CONSTRAINTS - A menu of the initial design variables and constraints will be displayed for an existing analysis subprogram. If this is a new analysis subprogram, the user will be asked to enter the initial values of the design variables and any side constraints on the design variables.

A hard copy printout of the optimizer control parameters, design variables and side constraints will then be presented to the user to check to ensure that the above information has been entered correctly. The user then has the option to return to the input routine and make changes to the above information or to continue and optimize his design. If the user chooses to continue, OPCON will overlay the optimization program on OPCON beginning at line C2. The analyze subprogram specified will then be linked to the end of the DESOP program. The entire program will then be STORED under the program name OP. The OP program will then be loaded with

execution beginning at line OPT1. The storing and reloading process allows the OP program to be called and run as a separate program for debugging purposes. At the completion of the optimization process, the optimizer program will reload OPCON and return execution to the OPCON program. A menu will be displayed welcoming the user back to the OPCON program giving him the following options: to optimize again using the same ANALIZ subprogram, to optimize again using a different ANALIZ subprogram, or to terminate the program.

APPENDIX D

OPCON Program Listing

```
100  !    ****************************************************************
110  !    *                                                              *
120  !    *                          OPCON                               *
130  !    *                                                              *
140  !    *     THE EXECUTIVE PROGRAM FOR PERFORMING NUMERICAL           *
150  !    *         OPTIMIMIZATION ON THE HP 9845A COMPUTER              *
160  !    *                                                              *
170  !    *          by  W. B. COLE                                      *
180  !    *          NAVAL POSTGRADUATE SCHOOL, MONTEREY, CALIFORNIA     *
190  !    *          1980                                                *
200  !    ****************************************************************
210  !
220  !
230  !
240  !    FUNCTIONS :
250  !       1.  INPUT AND STORAGE OF CONTROL PARAMETERS FOR THE OPTIMIZERS USED
260  !       2.  INPUT AND STORAGE OF THE DESIGN VARIABLES AND SIDE CONSTRAINTS
270  !           ON THE OPTIMIZERS USED
280  !       3.  CONTROL OF THE OPTIMIZATION PROCESS THROUGH THE USE OF OVERLAYS
290  !
300  ! ------------------------------------------------------------------------
310  ! NOTE: OPCON was written to control multiple optimization programs.  At
320  !       the present time DESOP is the only optimizer working.  A Feasible
330  !       Directions optimizer has been partialy written and debugged.  At
340  !       the present time OPCON has been written to accept the control
350  !       parameters for the Feasible Direction optimization program.
360  ! ------------------------------------------------------------------------
370  !                  ----- CONTROL SECTION -----
380  ! ------------------------------------------------------------------------
390  COM Globcm(50),P$,D$,Analiz$
400  GOTO C7
```

```
410  ! ------------------------------------------------------------------------
420 C1:  !          ----- CONTROL FOR SUMT OPTIMIZER -----
430  ! ------------------------------------------------------------------------
440  PRINTER IS 16
450  PRINT PAGE
460  PRINT LIN(5),TAB(10)," LINKING SUMT OPTIMIZER AND ";Analiz$;" ROUTINE."
470  LINK Op$,C2
480  LINK Analiz$,Analiz1
490  RE-STORE "OP"
500  LOAD "OP",Opt1
510  ! ------------------------------------------------------------------------
520 C2:  !          ----- ATTACHMENT POINT FOR OPTIMIZER -----
530  ! ------------------------------------------------------------------------
540  !
550  ! ------------------------------------------------------------------------
560 C7:  !                  ----- INTRODUCTION -----
570  ! ------------------------------------------------------------------------
580  PRINTER IS 16
590  PRINT PAGE
600  PRINT "                              ***   OPCON   ***"
610  PRINT LIN(2),"OPCON IS THE CONTROL PROGRAM FOR PERFORMING NUMERICAL OPTIM
IZATION ON "
620  PRINT LIN(1),"THE HP 9845 DESKTOP COMPUTER."
630  PRINT LIN(2),"THE FUNCTIONS OF OPCON ARE :"
640  PRINT LIN(1),"             1.   THE INPUT AND STORAGE OF CONTROL PARAMETERS.
"
650  PRINT "            2.   THE INPUT AND STORAGE OF DESIGN VARIABLES."
660  PRINT "            3.   CONTROL OF THE OPTIMIZATION PROCESS THROUGH PROGRAM
 OVERLAYS."
670  DISP " PRESS  `CONT´ "
680  PAUSE
690  GOTO C10
```

```
700  ! --------------------------------------------------------------------
710 C8:  !              ----- RETURN FROM OPTIMIZATION -----
720  ! --------------------------------------------------------------------
730  PRINTER IS 16
740  PRINT PAGE
750  PRINT "                         *** WELCOME BACK ***  "
760  PRINT LIN(1),"WELCOME BACK TO `OPCON'.   YOU NOW HAVE SEVERAL AVAILABLE OPT
IONS."
770  PRINT LIN(1),"      1.  OPTIMIZE THE ";Analiz$;" FUNCTION."
780  PRINT LIN(1),"      2.  OPTIMIZE A DIFFERENT ANALIZE FUNCTION."
790  PRINT LIN(1),"      3.  TERMINATE THE PROGRAM."
800  INPUT " INPUT YOUR CHOICE : 1,2 OR 3 .",Ch
810  IF Ch=1 THEN Prog1=1
820  IF Ch=1 THEN GOTO C13
830  IF Ch=2 THEN Prog1=0
840  IF Ch=2 THEN GOTO C12
850  IF Ch=3 THEN PRINT PAGE
860  IF Ch=3 THEN PRINT "                                  END OF PROGRAM"
870  IF Ch=3 THEN GOTO 299
880  GOSUB Err
890  GOTO C8
900  ! --------------------------------------------------------------------
910 C9:  !        ----- RETURN FROM END OF INPUT ROUTINE -----
920  ! --------------------------------------------------------------------
930  PRINTER IS 16
940  PRINT PAGE
950  PRINT "DO YOU STILL WISH TO OPTIMIZE THE ";Analiz$;" FUNCTION?"
960  INPUT " ( 1 - YES  :  0 - NO )",Ch
970  IF Ch=0 THEN C10
980  IF Ch=1 THEN Para9
990  GOSUB Err
1000 GOTO C9
```

```
1010 ! -------------------------------------------------------------------
-
1020 C10:   !        -----  CONTROL PARAMETER INPUT -----
1030 ! -------------------------------------------------------------------
-
1040 DIM Lmin(4),Ladd(4),P(50,4),Param$(50),X(30),Vlb(30),Vub(30)
1050 DIM S$(40),Params(50)
1060   PRINTER IS 16 ! (CRT)
1070   PRINT PAGE
1080   ! ---------------------------------------------------------------
1090 C11: !              ----- NEW OR EXISTING PROGRAM ? -----
1100   ! ---------------------------------------------------------------
1110   PRINT LIN(3),TAB(17),"IS THIS A NEW OR EXISTING PROGRAM ?"
1120   INPUT "     INPUT : ( NEW = 0 , EXISTING = 1 ) ",Prog1
1130   ! ---------------------------------------------------------------
1140 C12: !               ----- NAME THE ANALIZE PROGRAM -----
1150   ! ---------------------------------------------------------------
1160   PRINT PAGE
1170   PRINT LIN(3),TAB(10),"WHAT IS THE NAME OF THE ANALIZE PROGRAM ?"
1180   INPUT "   INPUT THE NAME : ",Analiz$
1190   DISP "CREATING NEW FILES FOR ";Analiz$
1200   P$="P"                               ! ASSIGN PARAMETER AND DESIGN VARIABLE
1210   D$="D"                               ! FILE NAMES TO MATCH THE ANALIZE
1220   P$[2,6]=Analiz$                      ! ROUTINE.
1230   D$[2,6]=Analiz$
1240   IF Prog1=1 THEN GOTO 1290            ! IF THIS IS A NEW PROGRAM, CREATE THE
1250   ON ERROR GOTO Err1
1260   CREATE P$,4
1270   CREATE D$,3
1280   OFF ERROR
1290   PRINT PAGE
```

```
1300   ! ------------------------------------------------------------------------
1310 C13: !                 ----- OPTIMIZER SELECTION -----
1320   ! ------------------------------------------------------------------------
1330   PRINT LIN(3),TAB(5),"WHICH OF THE FOLLOWING OPTIMIZERS DO YOU WISH TO US
E ?"
1340   PRINT LIN(1),TAB(5),"1.  SEQUENTIAL UNCONSTRAINED OPTIMIZATION. DESOP"
1350   ! PRINT LIN(1),TAB(5),"2.  FEASIBLE-DIRECTION OPTIMIZATION."
1360   PRINT LIN(3),"  AT THE PRESENT TIME DESOP IS THE ONLY OPTIMIZATION PROG
RAM AVAILABLE."
1370   INPUT " INPUT THE OPTIMIZER NUMBER : ",Prog2
1380   IF Prog2=1 THEN Op$="DESOP"
1390   IF Prog2=2 THEN Op$="OPT-2"
1400   IF Prog1=1 THEN GOTO Para10
1410   ! ------------------------------------------------------------------------
1420 C14: !            ----- INPUT INITIAL VALUE FOR NDV -----
1430   ! ------------------------------------------------------------------------
1440   Param$(1)="NDV"
1450   P(1,1)=1
1460   P(1,2)=30
1470   P(1,3)=0
1480   PRINT PAGE
1490   PRINT LIN(3),TAB(5)," INPUT THE NUMBER OF DESIGN VARIABLES ( NDV ) "
1500   PRINT TAB(5),"MIN = ",P(1,1)
1510   PRINT TAB(5),"MAX = ",P(1,2)
1520   INPUT " Ndv = ? ",P(1,4)
1530   Ndv=P(1,4)
1540   IF (P(1,4)>=P(1,1)) AND (P(1,4)<=P(1,2)) THEN Para10
1550   GOSUB Err
1560   GOTO 1480
1570 Para10: ! CONTINUE
```

```
1580  ! ---------------------------------------------------------------------------
1590  !                  ----- CONTROL PARAMETER DATA -----
1600  ! ---------------------------------------------------------------------------
1610  DATA 2,10
1620  DATA NCON,0,20,0
1630  DATA IPRINT,-5,6,0
1640  DATA DELFUN,0.0,0.1,0.001
1650  DATA DABFUN,0.0,0.1,0.001
1660  DATA ITMAX,1,100,20
1670  DATA ICNDIR,2,21,5
1680  DATA FDCH,1E-5 ,0.1,0.001
1690  DATA FDCHM,1E-6 ,0.1,0.0001
1700  DATA ABOBJ1,0.01,0.5,0.1
1710  DATA ALPHAX,0.01,0.5,0.1
1720  ! ---------------------------------------------------------------------------
1730  !             -----  OPTIMIZER #2 CONTROL PARAMETERS -----
1740  ! ---------------------------------------------------------------------------
1750    DATA 20,10
1760    DATA IPDBG,0,1,0
1770    DATA CT,0,1,0.1
1780    DATA CTMIN,-0.01,0,-0.004
1790    DATA CTL,0,1,0.1
1800    DATA CTLMIN,-0.1,0,-0.001
1810    DATA PHI,1,100,5
1820    DATA THETA,0,10,1
1830    DATA ITRM,1,5,3
1840    DATA CG,0,1,0
1850    DATA NCF,0,40,0
```

```
1860    ! ----------------------------------------------------------------------------
1870    !                     ----- OPTIMIZER #1 CONTROL PARAMETERS -----
1880    ! ----------------------------------------------------------------------------
1890    DATA 40,5
1900    DATA IRMAX,1,30,10                      ! MAXIMUM NUMBER OF PENALTY ITERATIONS
1910    DATA RZ,.0001,9999,2                    ! INITIAL PENALTY FUNCTION
1920    DATA RMULT,0.001,1000,2                 ! PENALTY FUNCTION MULTIPLIER
1930    DATA EXPG,1,10,2                        ! PENALTY FUNCTION EXPONENT
1940    DATA NSCAL,0,1,1                        ! DESIGN VARIABLE SCALING 1-YES 0-NO
1950    FOR K=1 TO 3                            ! READ DATA
1960    READ Lmin(K),Ladd(K)
1970    FOR I=Lmin(K) TO Lmin(K)+Ladd(K)-1
1980    READ Param$(I),P(I,1),P(I,2),P(I,3)
1990    NEXT I
2000    NEXT K
2010    IF Progl=1 THEN GOTO P3       ! (CONTINUE)
2020    FOR I=2 TO 50
2030    P(I,4)=P(I,3)
2040    NEXT I
2050    ! ----------------------------------------------------------------------------
2060 P3:  !       ----- IF EXISTING PROGRAM - READ EXISTING VALUES -----
2070    ! ----------------------------------------------------------------------------
2080    IF Progl=0 THEN Para9                     ! IF NOT CONTINUE
2090    Param$(1)="NDV"
2100    DISP "READING EXISTING CONTROL PARAMETERS AND DESIGN VARIABLES."
2110    ASSIGN #1 TO P$
2120    ASSIGN #2 TO D$
2130    FOR J=1 TO 3
2140    READ #1,J
2150    K=Lmin(J)-(J=1)
2160    K1=K+Ladd(J)-1+(J=1)
2170    FOR I=K TO K1
2180    READ #1;P(I,4)
2190    NEXT I
2200    NEXT J
2210    Ndv=P(1,4)
```

```
2220   ! -------------------------------------------------------------------------
2230   !                ----- READ EXISTING DESIGN VARIABLES -----
2240   ! -------------------------------------------------------------------------
2250   READ #2,1
2260   FOR I=1 TO Ndv
2270   READ #2;X(I)
2280   NEXT I
2290   READ #2,2
2300   FOR I=1 TO Ndv
2310   READ #2;Vlb(I),Vub(I)
2320   NEXT I
2330 Para9: !  CONTINUE
2340   ! -------------------------------------------------------------------------
2350   !                          ----- INPUT CHECK -----
2360   ! -------------------------------------------------------------------------
2370    K=1
2380   PRINT PAGE
2390   PRINT TAB(10),"INPUT CHECK OF COMMON CONTROL PARAMETERS"
2400   PRINT LIN(1)
2410   PRINT "REF #  PARAMETER     MINIMUM       MAXIMUM    RECOMMENDED
PRESENT"
2420   IMAGE 4X,"1",3X,6A,4X,M4D.4D,4X,M4D.4D,8X,"******",4X,M4D.4D
2430   PRINT USING 2420;Param$(1),P(1,1),P(1,2),P(1,4)
2440   IMAGE 4X,"2",3X,6A,4X,M4D.4D,4X,M4D.4D,8X,"******",4X,M4D.4D
2450   PRINT USING 2440;Param$(2),P(2,1),P(2,2),P(2,4)
2460   FOR I=3 TO Ladd(1)+1
2470   IMAGE 3X,DD,3X,6A,4(4X,M4D.4D)
2480   PRINT USING 2470;I,Param$(I),P(I,1),P(I,2),P(I,3),P(I,4)
2490   NEXT I
2500   PRINT LIN(1)," NOTE : IPRINT = 6 IS DEBUG MODE, THERE IS NO HARDCOPY OU
TPUT ! "
2510   IF Ie=1 THEN GOTO 2560
2520   INPUT "DO YOU WISH TO MAKE ANY CHANGES?  ENTER REF# OR 0 FOR NONE.",Ch
2530   IF Ch=0 THEN Para11
2540   IF (Ch<0) OR (Ch>Ladd(K)+1) THEN GOSUB Err
2550   IF (Ch<0) OR (Ch>Ladd(K)+1) THEN GOTO 2380
```

```
2560   DISP "INPUT THE CHANGE TO ";Param$(Ch);
2570   INPUT Pch
2580   IF (Pch>=P(Ch,1)) AND (Pch<=P(Ch,2)) THEN Para13
2590   GOSUB Err
2600   Ie=1
2610   GOTO 2380
2620 Para13:  P(Ch,4)=Pch
2630   Ie=0
2640   GOTO 2380
2650 Para12:  ! ( INPUT ERROR )
2660   GOTO 2380
2670   ! -------------------------------------------------------------------------------
2680 Para11:  !       ----- INPUT CHECK  FEASIBLE-DIRECTION -----
2690   ! -------------------------------------------------------------------------------
2700   GOTO Para15                         ! LOOP AROUND FEASIBLE DIRECTION
2710   PRINT PAGE
2720   IF Prog2<>1 THEN Para15
2730   K=2
2740   PRINT TAB(5),"INPUT CHECK   -   USABLE-FEASABLE CONTROL PARAMETERS"
2750   PRINT LIN(1)
2760   PRINT "REF #  PARAMETER    MINIMUM       MAXIMUM    RECOMMENDED
PRESENT"
2770   FOR I=Lmin(K) TO Lmin(K)+Ladd(K)
2780   IMAGE 3X,DD,3X,6A,4(4X,M4D.4D)
2790   PRINT USING 2780;I,Param$(I),P(I,1),P(I,2),P(I,3),P(I,4)
2800   NEXT I
2810   IF Ie=1 THEN Para14
2820   PRINT LIN(2),"DO YOU WISH TO MAKE ANY CHANGES ?"
2830   INPUT " ( INPUT : REF #  -OR-  0  FOR NONE )",Ch
2840   IF Ch=0 THEN Para15
2850   IF (Ch<Lmin(K)) OR (Ch>Lmin(K)+Ladd(K)) THEN GOSUB Err
2860   IF (Ch<Lmin(K)) OR (Ch>Lmin(K)+Ladd(K)) THEN GOTO 2710
2870 Para14:  DISP "INPUT THE CHANGE TO ";Param$(Ch);
2880   INPUT Pch
2890   IF (Pch>=P(Ch,1)) AND (Pch<=P(Ch,2)) THEN Para17
2900   GOSUB Err
2910   Ie=1
```

```
2920   GOTO 2710
2930 Para17: P(Ch,4)=Pch
2940   Ie=0
2950   GOTO 2710
2960   ! ------------------------------------------------------------------------
2970 Para15: !              ----- INPUT CHECK  DESOP -----
2980   ! ------------------------------------------------------------------------
2990   PRINT PAGE
3000   IF Prog2<>2 THEN Para25
3010   K=3
3020   PRINT TAB(15),"INPUT CHECK  -  `SUMT' CONTROL PARAMETERS"
3030   PRINT LIN(1)
3040   PRINT "REF #  PARAMETER     MINIMUM       MAXIMUM    RECOMMENDED
     PRESENT"
3050   FOR I=Lmin(K) TO Lmin(K)+Ladd(K)-1
3060   IMAGE 3X,DD,3X,6A,4(4X,M4D.4D)
3070   PRINT USING 2780;I,Param$(I),P(I,1),P(I,2),P(I,3),P(I,4)
3080   NEXT I
3090   IF Ie=1 THEN Para14
3100   PRINT LIN(2),"DO YOU WISH TO MAKE ANY CHANGES ?"
3110   INPUT " ( INPUT : REF #  -OR-  0  FOR NONE )",Ch
3120   IF Ch=0 THEN Para25
3130   IF (Ch<Lmin(K)) OR (Ch>Lmin(K)+Ladd(K)) THEN GOSUB Err
3140   IF (Ch<Lmin(K)) OR (Ch>Lmin(K)+Ladd(K)) THEN GOTO 2990
3150 Para24:  PRINT LIN(1),"INPUT THE CHANGE TO ";Param$(Ch)
3160   INPUT Pch
3170   IF (Pch>=P(Ch,1)) AND (Pch<=P(Ch,2)) THEN Para27
3180   Ie=1
3190   GOTO 2990
3200 Para27: P(Ch,4)=Pch
3210   Ie=0
3220   GOTO 2990
3230 Para25: ! CONTINUE
3240   IF Prog1=1 THEN Sidein6           !EXISTING PROGRAM GOTO INPUT CHECK
3250   I1=0
```

```
3260   PRINT PAGE
3270   PRINT TAB(15),"INPUT THE INITIAL VALUES FOR THE DESIGN VARIABLES"
3280   PRINT LIN(1)
3290   GOSUB Sidein
3300   PRINT PAGE
3310   PRINT "END OF INPUT ROUTINE"
3320   ! -----------------------------------------------------------------------
3330 Sav:  !                    ------ SAVE INPUTS ------
3340   ! -----------------------------------------------------------------------
3350   PRINT PAGE
3360   Iprint=P(3,4)
3370   IF Iprint=6 THEN S1
3380   PRINTER IS 0
3390   PRINT LIN(3)
3400   PRINT "-----------------------------------------------------------------------
-----------"
3410   PRINT LIN(3)
3420   PRINT "ANALIZE PROGRAM :    ";Analiz$
3430   IF Prog2=1 THEN PRINT LIN(2),"OPTIMIZER USED :     DESOP"
3440   ! IF Prog2=2 THEN PRINT LIN(2),"OPTIMIZER USED :     FEASIBLE-DIRECTION
3450   PRINT LIN(2),"INPUT FOR OPTIMIZATION :"
3460   PRINT LIN(1)
3470 S1:    DISP "SAVING INPUT PARAMETERS."
3480   ASSIGN #1 TO P$                          !PUT PARAMS INTO MASS STORAGE
3490   ASSIGN #2 TO D$
3500   PRINT #1,4;Ladd(1)+1,Ladd(2),Ladd(3)
3510   FOR J=1 TO 3                             ! J = 1 COMMON CONTROL PARAMS
3520   READ #1,J                                ! J = 2 U/F CONTROL PARAMETERS
3530   K=Lmin(J)-(J=1)                          ! J = 3 SUMT CONTROL PARAMS
3540   K1=K+Ladd(J)-1+(J=1)
3550   FOR I=K TO K1
3560   PRINT Param$(I);"=";P(I,4),
3570   PRINT #1;P(I,4)
3580   NEXT I
3590   PRINT #1;END
3600   NEXT J
```

```
3610   ! ------------------------------------------------------------------
3620   !                  ----- SAVE THE DESIGN VARIABLES -----
3630   ! ------------------------------------------------------------------
3640   DISP "SAVING DESIGN VARIABLES"
3650   PRINT LIN(1)
3660   READ #2,1
3670   FOR I=1 TO Ndv
3680   PRINT "X(";I;") = ";X(I),
3690   PRINT #2;X(I)
3700   NEXT I
3710   PRINT #2;END
3720   READ #2,2
3730   PRINT LIN(1)
3740   FOR I=1 TO Ndv                         ! SAVE UPPER AND LOWER BOUNDS
3750   PRINT "VLB("&VAL$(I)&") = ";Vlb(I);TAB(40),"VUB("&VAL$(I)&") = ";Vub(I)
3760   PRINT #2;Vlb(I),Vub(I)
3770   NEXT I
3780   PRINT #2;END
3790   DISP "CHECK YOUR INPUT  -  PRESS `CONT' TO CONTINUE."
3800   BEEP
3810   PAUSE
3820   GOTO P20
3830   ! ------------------------------------------------------------------
3840 Sidein:  !           ----- DESIGN VARIABLE INPUT -----
3850   ! ------------------------------------------------------------------
3860   Addcon=0
3870   FOR I=1 TO Ndv
3880   GOSUB Sidein1
3890   NEXT I
3900   I=0
3910 Sidein2: INPUT "DO YOU WISH TO MAKLE ANY CORRECTIONS? ( 0 FOR NO ; VAR# FOR
 YES )",I
3920   IF I=0 THEN Sidein3
3930   IF (I>0) AND (I<=P(1,4)) THEN GOSUB Sidein1
3940   IF (I>0) AND (I<=P(1,4)) THEN GOTO Sidein2
3950   GOSUB Err
3960   GOTO Sidein2
```

```
3970   ! ----------------------------------------------------------------------
3980 Sidein1: !      ----- INPUT INITIAL DESIGN VARIABLES -----
3990   ! ----------------------------------------------------------------------
4000   DISP "INPUT INITIAL X("&VAL$(I)&"),VLB,VUB : ( USE N FOR NO LOWER OR UPPE
R BOUND )";
4010   INPUT X(I),L$,U$
4020   IF L$="N" THEN Vlb(I)=-1E50
4030   IF L$<>"N" THEN Vlb(I)=VAL(L$)
4040   IF L$="N" THEN L$="NONE"
4050   IF U$="N" THEN Vub(I)=1E50
4060   IF U$<>"N" THEN Vub(I)=VAL(U$)
4070   IF U$="N" THEN U$="NONE"
4080   PRINT " X("&VAL$(I)&")=";X(I),"  VLB=";L$,"  VUB=";U$
4090   RETURN
4100 Sidein3: J=Ncon+1
4110   FOR I=1 TO P(1,4)
4120   IF Vlb(I)<=-1E49 THEN Sidein4
4130   Wlincon(J)=-I
4140   J=J+1
4150   Addcon=Addcon+1
4160 Sidein4:NEXT I
4170   FOR I=1 TO P(1,4)
4180   IF Vub(I)>=1E49 THEN Sidein5
4190   Wlincon(J)=I
4200   J=J+1
4210   Addcon=Addcon+1
4220 Sidein5:NEXT I
4230   RETURN
```

```
4240   ! --------------------------------------------------------------------------
4250 Sidein6: !        ----- INPUT CHECK FOR EXISTING PROGRAM -----
4260   ! --------------------------------------------------------------------------
4270   PRINT PAGE
4280   PRINT TAB(10),"INPUT CHECK  -  INITIAL VALUES FOR THE DESIGN VARIABLES"
4290   PRINT LIN(2)
4300   Z=Ndv
4310   IF Z>15 THEN Z=15
4320   FOR I=1 TO Z
4330   L$=VAL$(Vlb(I))
4340   IF Vlb(I)=-1E50 THEN L$="NONE"
4350   U$=VAL$(Vub(I))
4360   IF Vub(I)=1E50 THEN U$="NONE"
4370   PRINT I;"   X("&VAL$(I)&") = ";X(I),"VLB("&VAL$(I)&") = ";L$,"VUB("&VAL$(
I)&") = ";U$
4380   NEXT I
4390   DISP "DO YOU WISH TO MAKE ANY CHANGES ( REF# OR 0 FOR NONE ) ";
4400   INPUT Ch
4410   IF Ch=0 THEN Sidein8
4420   IF (Ch>0) AND (Ch<=Z) THEN GOSUB Sidein7
4430   IF (Ch>0) AND (Ch<=Z) THEN Sidein6
4440   GOSUB Err
4450   GOTO Sidein6                              ! TRY AGAIN
```

```
4460   ! ----------------------------------------------------------------------------
4470 Sidein7: !          ----- ENTER CHANGE TO DESIGN VARIABLE -----
4480   ! ----------------------------------------------------------------------------
4490   DISP "INPUT CHANGE TO X("&VAL$(Ch)&"),VLB("&VAL$(Ch)&"),VUB("&VAL$(Ch)&")
";
4500   INPUT X(Ch),L$,U$
4510   IF L$="N" THEN Vlb(Ch)=-1E50              ! SET UPPER AND LOWER LIMITS TO
4520   IF U$="N" THEN Vub(Ch)=1E50               ! +1E99 IF NOT SPECIFIED
4530   IF L$<>"N" THEN Vlb(Ch)=VAL(L$)
4540   IF U$<>"N" THEN Vub(Ch)=VAL(U$)
4550   RETURN                                    ! RETURN TO ENTER CHANGES
4560 Sidein8:  ! CONTINUE WITH INPUT CHECK
4570   IF Z<=15 THEN Sav
4580   Z=16
4590   PRINT PAGE
4600   PRINT TAB(10),"INPUT CHECK CONT.  -  INITIAL VALUES FOR THE DESIGN VARIA
BLES"
4610   PRINT LIN(2)
4620   FOR I=Z TO Ndv
4630   PRINT I;"   X("&VAL$(I)&") = ";X(I),"VLB("&VAL$(I)&") = ";Vlb(I),"VUB("&V
AL$(I)&") = ";Vub(I)
4640   NEXT I
4650   DISP "(DO YOU WISH TO MAKE ANY CHANGES ( REF# OR 0 FOR NONE )";
4660   INPUT Ch
4670   IF Ch=0 THEN Sav                        ! GOTO PRINT INPUT ROUTINE
4680   IF (Ch>16) AND (Ch<=Ndv) THEN GOSUB Sidein7
4690   IF (Ch>16) AND (Ch<=Ndv) THEN GOTO 4650
4700   GOSUB Err
4710   GOTO 4650
4720   ! ----------------------------------------------------------------------------
4730 Err: !          ----- ERROR ROUTINE FOR ILLEGAL ENTRY -----
4740   ! ----------------------------------------------------------------------------
4750   PRINT PAGE
4760   BEEP
4770   PRINT LIN(10),TAB(10),"*****  INPUT OUT OF RANGE  -  TRY AGAIN  *****"
4780   WAIT 2000
4790   RETURN
```

```
4800   ! ----------------------------------------------------------------------
4810 Err1:  !        ----- ERROR ROUTINE FOR EXISTING PROGRAM -----
4820   ! ----------------------------------------------------------------------
4830   PRINTER IS 16
4840   PRINT PAGE
4850   BEEP
4860   PRINT LIN(10),TAB(20),"***** ";Analiz$;" IS AN EXISTING PROGRAM *****"
4870   WAIT 5000
4880   OFF ERROR
4890   GOTO C10
4900   ! ----------------------------------------------------------------------
4910 Err2:  !         ----- ERROR ROUTINE FOR EXISTING DATA -----
4920   ! ----------------------------------------------------------------------
4930   PRINT PAGE
4940   BEEP
4950   PRINT TAB(15),"*** READ ERROR ON EXISTING PROGRAM ***"
4960   PRINT LIN(2)
4970   PRINT "A READ ERROR HAS OCCURED WHILE ATTEMPTING TO READ EXISTING DATA."
4980   PRINT "THERE ARE SEVERAL POSSIBLE CAUSES FOR THIS TO HAVE HAPPENED."
4990   PRINT LIN(1),"     1.  A MISTAKE WAS MADE WHILE INPUTING DATA.  IF SO RES
TART"
5000   PRINT "     THE PROGRAM BY PRESSING THE RUN KEY.  "
5010   PRINT LIN(1),"     2.  THE DATA FILES NO LONGER EXIST FOR ";Analiz$;" OR
HAVE AN ERROR "
5020   PRINT "     IN THEM.   IF THIS IS THE CASE THEY MUST BE ERASED AND THE PR
OGRAM "
5030   PRINT "     RESTRATED AS A NEW PROGRAM.  PRESS THE CONT KEY TO ERASE THE
EXISTING"
5040   PRINT "     DATA FILES FOR ";Analiz$;" AND RESTART THE PROBLEM."
5050   PAUSE
5060   PURGE P$
5070   PURGE D$
5080   OFF ERROR
5090   GOTO C10
```

```
5100   ! --------------------------------------------------------------------
5110 P20:  !                  ----- LAST CHANCE -----
5120   ! --------------------------------------------------------------------
5130   PRINTER IS 16
5140   PRINT PAGE
5150   PRINT TAB(10),"LAST CHANCE TO CHANGE YOUR MIND BEFORE OPTIMIZING"
5160   PRINT LIN(2),TAB(5),"1.  RETURN TO THE CONTROL PARAMETER AND DESIGN VARIA
BLE INPUT ROUTINE."
5170   PRINT LIN(2),TAB(5),"2.  PROCEDE TO OPTIMIZE THE ";Analiz$;" FUNCTION."
5180   INPUT "INPUT YOUR DECISION :",Ch
5190   IF Ch=1 THEN C9
5200   IF (Ch=2) AND (Prog2=1) THEN C1       ! OVERLAY THE DESOP PROGRAM
5210   IF (Ch<1) AND (Ch>2) THEN GOSUB Err
5220   GOTO P20
5230   ! --------------------------------------------------------------------
5240 Z99: !                       ----- END -----
5250   ! --------------------------------------------------------------------
5260   END
```

APPENDIX E

DESOP Program Listing

In the development of the DESOP program, Refs. 2, 7, 8, 9, and 10 were used.

```
100   !     ****************************************************************
105   !     *                                                              *
110   !     *                           DESOP                              *
115   !     *                                                              *
120   !     *      DESKTOP SEQUENTIAL UNCONSTRAINED MINIMIZATION           *
125   !     *        TECHNIQUE OPTIMIZATION PROGRAM                        *
130   !     *                                                              *
135   !     *        by  W. B. COLE                                        *
140   !     *        NAVAL POSTGRADUATE SCHOOL, MONTEREY, CALIFORNIA       *
145   !     *        1980                                                  *
150   !     *                                                              *
155   !     ****************************************************************
160   !
165   !     DATE OF LAST REVISION : 8/29/80
170   !     PROGRAM FEATURES :
175   !     1. SEARCH METHODS:
180   !            A. STEEPEST DECENT
185   !            B. FLETCHER-REEVES
190   !            C. BOTH OF WHICH ARE NORMALIZED
195   !     2. ALPHA BOUND - UTILIZES THE GOLDEN SECTION TECHNIQUE TO LOCATE THE
200   !        FUNCTION MINIMUM.  THE PREDICTED MINIMUM IS TESTES AGAINST THE
205   !        ACTUAL VALUE OF OBJ AT THAT POINT.
210   !     3. A FOUR POINT CUBIC IS USED TO PREDICT THE MINIMUM.
215   !     4. IF THE CUBIC FIT FAILS THERE IS A QUADRATIC BACKUP.
220   !     5. ABOBJ1 IS ADJUSTED ON EACH ITERATION.
225   !     6. THE DESIGN VARIABLES ARE NORMALIZED
230   !
235 Opt1:   ! CONTROL POINT FOR TRANSFER OF PROGRAM CONTROL FROM INPUT ROUTINE TO
240        ! OPTIMIZER ROUTINE.
```

```
245   ! ------------------------------------------------------------------------
-
250   ! -----                    BEGIN MAIN PROGRAM                      ------
255   ! ------------------------------------------------------------------------
260 Opt2:   ! BEGIN
265   DIM X(30),G(30),Tmp(30),Params(50),Df(30),Vlb(30),Vub(30),Dfo(30),S(30)
270   DIM Xsav(30),Gg(30),C(10),Xscal(30),Scal(30)
275   ! ------------------------------------------------------------------------
280   !          READ CONTROL PARAMETERS AND INITIAL DESIGN VARIABLES
285   ! ------------------------------------------------------------------------
290   DISP "READING CONTROL PARAMETERS AND DESIGN VARIABLES."
295   ASSIGN #1 TO P$
300   ASSIGN #2 TO D$
305   READ #1,1
310   READ #1;Ndv,Ncon,Iprint,Delfun,Dabfun,Itmax,Icndir,Fdch,Fdchm,Abobj1,Alpha
x
315   READ #1,3
320   READ #1;Irmax,Rz,Rmult,Expg,Nscal
325   READ #2,1
330   FOR I=1 TO Ndv
335   READ #2;X(I)
340   NEXT I
345   READ #2,2
350   FOR I=1 TO Ndv
355   READ #2;Vlb(I),Vub(I)
360   NEXT I
365   Niter=0                               ! INITIALIZE PROGRAM PARAMETERS
370   Ncal=0
375   Ncc=0
380   C(2)=Iprint
385   IF Iprint<0 THEN PRINTER IS 0
390   IF Iprint>=0 THEN PRINTER IS 16
395   IF Ncon=0 THEN Irmax=1
```

```
400   FOR I=1 TO Ndv
405   Scal(I)=X(I)                                  ! SCALE THE DESIGN VARIABLES
410   IF Scal(I)<1 THEN Scal(I)=1
415   IF Nscal=0 THEN Scal(I)=1
420   Xscal(I)=X(I)/Scal(I)
425   Dfo(I)=0
430   NEXT I
435   ! ----------------------------------------------------------------------
440   !                    ----- INITIAL DESIGN -----
445   ! ----------------------------------------------------------------------
450   DISP "EVALUATING THE OBJECTIVE AND CONSTRAINT FUNCTIONS AT X(0)."
455   Icalc=1                                       ! FIRST AND LAST FLAG
460   C(1)=Icalc
465   CALL Analiz(X(*),G(*),Gg(*),C(*),Obj)
470   Ncal=Ncal+1
475   IF Ncon>0 THEN GOSUB Penalize
480   Icalc=2
485   Nscal=0                                       ! INITIALIZE SCALING COUNTER
490   C(1)=Icalc
495   !
500   Objsav=Obj
505   PRINTER IS 0
510   IF Iprint=6 THEN PRINTER IS 16
515   IF Iprint=6 THEN PRINT PAGE
520   IF Iprint=6 THEN GOTO 540
525   PRINT LIN(2)
530   PRINT "-----------------------------------------------------------------------
----------"
535   PRINT LIN(3)
540   PRINT "                                       INITIAL DESIGN   "
545   PRINT LIN(2),"DESIGN VARIABLES : "
550   FOR I=1 TO Ndv
555   PRINT "X("&VAL$(I)&")=";X(I),
560   NEXT I
565   IF Ncon<=0 THEN GOTO 590
```

```
570   PRINT LIN(2),"CONSTRAINTS : "
575   FOR I=1 TO Ncon
580   PRINT "G("&VAL$(I)&")=";G(I),
585   NEXT I
590   PRINT LIN(2),"OBJECTIVE FUNCTION :"
595   PRINT "OBJ=";Obj,"OBJA=";Objа
600   IF Iprint>=0 THEN PRINTER IS 16
605   DISP "PRESS `CONT' TO CONTINUE"
610   DISP "OPTIMIZER RUNNING"
615 O1:  ! CONTINUE
620   ! ----------------------------------------------------------------------
625   !          ----- OUTER LOOP FOR PENALTY FUNCTION -----
630   ! ----------------------------------------------------------------------
635   FOR Itr=1 TO Irmax
640   Objsvr=Obj
645   Kount=0
650   Dfold2=1
655   ! ----------------------------------------------------------------------
660   !       ----- INNER LOOP, NO INCREASE IN PENALTY FUNCTION -----
665   ! ----------------------------------------------------------------------
670   FOR Itm=1 TO Itmax
675   Niter=Niter+1
680   Kount=Kount+1
685   IF Kscal<>0 THEN GOTO 720
690   FOR I=1 TO Ndv                      ! SCALE THE DESIGN VARIABLES
695   Scal(I)=Xscal(I)*Scal(I)
700   IF Scal(I)<1 THEN Scal(I)=1
705   IF Nscal=0 THEN Scal(I)=1
710   Xscal(I)=X(I)/Scal(I)
715   NEXT I
720   Kscal=Kscal+1
725   IF Kscal=Ndv+2 THEN Kscal=0
730   IF Iprint<0 THEN PRINTER IS 0
735   IF ABS(Iprint)>1 THEN PRINT LIN(2)
740   IF ABS(Iprint)>1 THEN PRINT "**********************************************
********************************"
```

```
745   IF ABS(Iprint)>1 THEN PRINT LIN(1),"ITERATION NUMBER : ";Niter
750   ! ---------------------------------------------------------------------
755   ! ----- EVALUATE THE GRADIENT OF THE OBJECTIVE FUNCTION AND ANY   -----
760   ! ----- ACTIVE CONSTRAINTS AT X(0)                                 -----
765   ! ---------------------------------------------------------------------
770   DISP "CALCULATING THE GRADIENTS"
775   GOSUB Grad
780   !
785   IF ABS(Iprint)>3 THEN PRINT,"GRADIENTS :"
790   FOR I=1 TO Ndv
795   IF ABS(Iprint)>3 THEN PRINT "  DF("&VAL$(I)&")=";Df(I),
800   NEXT I
805   DISP "FINDING THE SEARCH DIRECTION"
810   ! ---------------------------------------------------------------------
815 O5:  !            ----- FIND THE SEARCH DIRECTION -----
820   ! ---------------------------------------------------------------------
825   Df2=0                                  ! INITIALIZE Df
830   Dfold2=0
835   FOR I=1 TO Ndv
840   Dfold2=Dfold2+Dfo(I)^2
845   Df2=Df2+Df(I)^2
850   NEXT I
855   IF Kount>1 THEN O2                     ! USE FLETCHER REEVES
860   ! -------------------------------------------------------------------------
865   !    ----- CALCULATION OF THE SEARCH DIRECTION BY STEEPEST DECENT -----
870   ! -------------------------------------------------------------------------
-
875   Fact=1E-3
880   FOR I=1 TO Ndv
885   S(I)=-Df(I)                           ! SEARCH DIRECTION
890   IF ABS(S(I))>Fact THEN Fact=ABS(S(I)) ! NORMALIZE SEARCH DIRECTION
895   Dfo(I)=Df(I)
900   NEXT I
905   FOR I=1 TO Ndv
910   S(I)=S(I)/Fact
915   NEXT I
920   IF ABS(Iprint)>2 THEN PRINT LIN(1),"SEARCH METHOD : STEEPEST DESCENT"
925   GOTO O3                               ! BRANCH AROUND FLETCHER REEVES
```

```
930 02:  ! CONTINUE
935   ! -----------------------------------------------------------------
940   !     -----CALCULATION OF SEARCH DIRECTION BY FLETCHER REEVES-----
945   ! -----------------------------------------------------------------
950   Beta=Df2*Fact/Dfold2
955   Fact=1E-3
960   FOR I=1 TO Ndv
965   S(I)=-Df(I)+Beta*S(I)                    ! SEARCH DIRECTION
970   IF ABS(S(I))>Fact THEN Fact=ABS(S(I))
975   Dfo(I)=Df(I)
980   NEXT I
985   FOR I=1 TO Ndv
990   S(I)=S(I)/Fact                           ! NORMALIZE THE SEARCH DIRECTION
995   NEXT I
1000  IF ABS(Iprint)>2 THEN PRINT LIN(1),"SEARCH METHOD : FLETCHER REEVES"
1005 03: ! CONTINUE
1010  IF Kount>=Icndir THEN Kount=0
1015  FOR I=1 TO Ndv
1020  IF ABS(Iprint)>2 THEN PRINT "    S("&VAL$(I)&")=";S(I),
1025  NEXT I
1030  ! ------------------------------------------------------------------
1035  !   -----CALCULATE THE SLOPE OF THE OBJECTIVE FUNCTION AT X(0) WITH-----
1040  !                   -----RESPECT TO ALPHA -----
1045  ! ------------------------------------------------------------------
1050  Dfdalp=0                                 ! OBJECTIVE FUNCTION SLOPE S(I)
1055  FOR I=1 TO Ndv
1060  Dfdalp=Dfdalp+Df(I)*S(I)
1065  NEXT I
1070  IF ABS(Dfdalp)<1E-20 THEN GOTO 099       ! IF SLOPE IS 0 THEN STOP
1075  IF ABS(Iprint)>2 THEN PRINT LIN(1),"DFDALP = ";Dfdalp
1080  IF Dfdalp>0 THEN Kount=0
1085  IF Dfdalp>0 THEN PRINT "THE OBJECTIVE FUNCTION HAS A POSITIVE SLOPE."
1090  IF Dfdalp>0 THEN PRINT "RESTART THE SEARCH USING STEEPEST DECENT."
1095  IF Dfdalp>0 THEN 05                     !START OVER WITH STEEPEST DECENT
```

```
1100  ! -----------------------------------------------------------------------
1105  !           ----- ESTIMATE ALPHA IN THE 1-D SEARCH -----
1110  ! -----------------------------------------------------------------------
-
1115  DISP "CALCULATING AN INITIAL ALPHA"
1120  !
1125  GOSUB Alpges
1126  IF Alp3<1E-12 THEN Alp3=1E-12
1130  IF ABS(Iprint)>2 THEN PRINT "ALPHA GUESS =";Alp3
1135  IF Iprint=6 THEN DISP "PRESS `CONT' TO CONTINUE."
1140  IF Iprint=6 THEN PAUSE
1145 04: ! CONTINUE
1150  ! -----------------------------------------------------------------------
1155  !           ----- CALCULATE ALPHA IN THE 1-D SEARCH -----
1160  ! -----------------------------------------------------------------------
1165  !
1170  DISP "FINDING ALPHA "
1175  !
1180  GOSUB Alpbnd
1185  !
1190  ! -----------------------------------------------------------------------
1195  !                    ----- ITERATION RESULTS -----
1200  ! ----------------------------------------------------------------------
1205  PRINT LIN(1),"RESULTS :"
1210  FOR I=1 TO Ndv
1215  X(I)=Xscal(I)*Scal(I)
1220  PRINT "X("&VAL$(I)&")=";X(I),
1225  NEXT I
1230  IF Ncon=0 THEN GOTO 1255
1235  PRINT LIN(1)
1240  FOR I=1 TO Ncon
1245  PRINT "G("&VAL$(I)&")=";G(I),
1250  NEXT I
1255  PRINT LIN(1),"OBJ = ";Obj,"OBJA = ";Obja
1260  DISP "PRESS `CONT' TO CONTINUE"
```

```
1265  ! ----------------------------------------------------------------------
1270  !                  ----- CHECK FOR CONVERGENCE -----
1275  ! ----------------------------------------------------------------------
1280  !
1285  GOSUB Convrg
1290  !
1295  IF ABS(Iprint)>2 THEN PRINT "DEL=";Del,"CONI=";Coni,LIN(1),"ABOBJ1=";Abobj
1
1300  IF Iprint=6 THEN PAUSE
1305  IF Coni>=2 THEN DISP "CONVERGENCE MET, INCREASE THE PENALTY FUNCTION."
1310  IF Coni>=2 THEN GOTO 1335          !IF CONVERGENCE IS SATISFIED TWICE
1315  Objsav=Obj                         !PROCEED TO INCREMENT RZ
1320  ! ----------------------------------------------------------------------
1325                            NEXT Itm
1330  ! ----------------------------------------------------------------------
1335  Rz=Rz*Rmult                        ! INCREASE THE PENALTY FUNCTION
1340  Rc=Rc+1                            ! COUNTER ON PENALTY INCREASES
1345  IF ABS(Iprint)>=2 THEN PRINT "RZ HAS BEEN INCREASED ";Rc;" TIMES TO ";Rz
1350  FOR I=1 TO Ndv
1355  X(I)=Xscal(I)*Scal(I)
1360  NEXT I
1365  CALL Analiz(X(*),G(*),Gg(*),C(*),Obj)
1370  Ncal=Ncal+1
1375  IF Ncon>0 THEN GOSUB Penalize
1380  GOSUB Convrg
1385  IF Coni>=3 THEN GOTO 099
1390  Coni=0
1395  ! ----------------------------------------------------------------------
1400                           NEXT Itr
1405  ! ----------------------------------------------------------------------
1410  !
1415  ! ----------------------------------------------------------------------
1420 099:   !          ----- OPTIMIZATION RESULTS -----
1425  ! ----------------------------------------------------------------------
1430  IF Iprint=6 THEN GOTO 1455
1435  PRINTER IS 0
1440  PRINT LIN(2)
```

```
1445  PRINT "------------------------------------------------------------------------
-----------"
1450  PRINT LIN(2)
1455  IF Iprint=6 THEN PRINT PAGE
1460  PRINT "                              OPTIMIZATION  RESULTS   "
1465  PRINT LIN(3),"FINAL DESIGN VARIABLES : "
1470  FOR I=1 TO Ndv
1475  X(I)=Xscal(I)*Scal(I)
1480  PRINT "X("&VAL$(I)&") = ";X(I),
1485  NEXT I
1490  IF Ncon<=0 THEN O6
1495  PRINT LIN(2),"FINAL CONSTRAINTS :"
1500  FOR I=1 TO Ncon
1505  PRINT "G("&VAL$(I)&")=";G(I),
1510  NEXT I
1515 O6:  PRINT LIN(2),"FINAL OBJECTIVE FUNCTION :"
1520  PRINT "OBJ=";Obj,"OBJA=";Obja
1525  PRINT LIN(2),"NUMBER OF ITERATIONS :";Niter
1530  PRINT LIN(2),"NUMBER OF TIMES ANALIZE CALLED :",Ncal
1535  PRINT LIN(2),"RZ HAS BEEN INCREASED ";Rc;" TIMES TO ";Rz
1540  ! PRINT ANY RESULTS SPECIFIED BY ANALIZE
1545  Icalc=3
1550  C(1)=Icalc
1555  CALL Analiz(X(*),G(*),Gg(*),C(*),Obj)
1560  DISP "END OF PROGRAM."
1565  LOAD "OPCON",C8
1570  END
```

```
1575  ! ************************************************************************
1580 Grad:  !                  ----- SUBROUTINE GRAD -----
1585  ! ************************************************************************
1590  !
1595  ! ------------------------------------------------------------------------
1600  !      -----STORE THE CURRENT VALUES OF THE ANALIZE SUBROUTINE -----
1605  ! ------------------------------------------------------------------------
1610  Objsav=Obj
1615  IF Ncon<=0 THEN GOTO G1
1620  FOR J=1 TO Ncon
1625  Tmp(J)=G(J)
1630  NEXT J
1635 G1: ! CONTINUE
1640  ! ------------------------------------------------------------------------
1645  !                ----- CALCULATE THE GRADIENTS -----
1650  ! ------------------------------------------------------------------------
1655  FOR J=1 TO Ndv
1660  Xsav=Xscal(J)
1665  Dx=ABS(Xsav)*Fdch
1670  IF Dx<Fdchm THEN Dx=Fdchm
1675  Xscal(J)=Xscal(J)+Dx
1680  ! ------------------------------------------------------------------------
1685  !          ----- CALCULATE THE FUNCTION AT X(J) + DX(J) -----
1690  ! ------------------------------------------------------------------------
1695  X(J)=Xscal(J)*Scal(J)                 ! CONVERT TO REAL X(*) TO ANALIZE
1700  CALL Analiz(X(*),G(*),Gg(*),C(*),Obj)
1705  Ncal=Ncal+1
1710  IF Ncon>0 THEN GOSUB Penalize
1715  !
1720  Df(J)=(Obj-Objsav)/Dx
```

```
1725  ! ----------------------------------------------------------------
1730  !     ----- RETURN X(I) AND OBJ TO THEIR ORIGINAL VALUES -----
1735  ! ----------------------------------------------------------------
1740  Xscal(J)=Xsav
1745  X(J)=Xscal(J)*Scal(J)
1750  NEXT J
1755  Obj=Objsav
1760  IF Ncon<=0 THEN GOTO G2
1765  FOR J=1 TO Ncon
1770  G(J)=Tmp(J)
1775  NEXT J
1780 G2: ! CONTINUE
1785  RETURN
1790  !
```

```
1795  ! *************************************************************************
1800 Alpges:  !          ----- SUBROUTINE ALPGES -----
1805  ! *************************************************************************
1810  !
1815  ! -------------------------------------------------------------------------
1820  !   CALCULATE AN INITIAL ESTIMATE FOR ALPHA IN THE 1-D SEARCH
1825  ! -------------------------------------------------------------------------
1830  Denom=ABS(Dfdalp)
1835  Anum=ABS(Obj)
1840  IF Denom<1E-5 THEN Denom=1E-5            ! PROTECT AGAINST DIVISION BY 0
1845  IF Anum<1 THEN Anum=1                    ! PROTECT AGAINST A SMALL NUMERATOR
1850  Alp3=Abobj1*Anum/Denom                   ! ALP3 = PRESENT UPPER BOUND
1855  FOR I=1 TO Ndv
1860  Axi=ABS(Xscal(I))
1865  IF Axi<1 THEN Axi=1
1870  Asi=ABS(S(I))
1875  IF Asi<1E-5 THEN Asi=1E-5
1880  Alp=Alphax*Axi/Asi
1885  IF Alp<Alp3 THEN Alp3=Alp
1890  NEXT I
```

```
1895  ! -------------------------------------------------------------------------
1900  !     -----  CHECK TO SEE IF ANY SIDE CONSTRAINTS ARE VIOLATED -----
1905  ! -------------------------------------------------------------------------
1910  FOR I=1 TO Ndv
1915  Vlscal(I)=Vlb(I)/Scal(I)
1920  Vuscal(I)=Vub(I)/Scal(I)
1925  IF ABS(Xscal(I)-Vlscal(I))>1E50 THEN Ag1
1930  IF ABS(S(I))<1E-20 THEN Ag1
1935  Alp=(Xscal(I)-Vlscal(I))/S(I)
1940  IF ABS(Alp)<ABS(Alp3) THEN Alp3=Alp
1945 Ag1: !
1950  IF ABS(Xscal(I)-Vuscal(I))>1E50 THEN Ag2
1955  IF ABS(S(I)) THEN Ag2
1960  Alp=(Vuscal(I)-Xscal(I))/S(I)
1965  IF ABS(Alp)<ABS(Alp3) THEN Alp3=Alp
1970 Ag2: !
1975  Alp3=ABS(Alp3)
1980  NEXT I
1985  !
1990  RETURN
1995  !
```

```
2000   ! ***********************************************************************
2005 Alpbnd:  !                  ----- SUBROUTINE ALPBND -----
2010   ! ***********************************************************************
2015   !
2020   ! -----------------------------------------------------------------------
2025   !                            ----- FIND ALPHA -----
2030   ! ----------------------------------------------------------------------
2035   !
2040   IF ABS(Iprint)>4 THEN PRINT "SUBROUTINE : ALPBND"
2045   IF ABS(Iprint)=5 THEN PRINT "DETERMINING THE UPPER AND LOWER BOUNDS ON AL
PHA."
2050   ! ----------------------------------------------------------------------
2055   !                      ----- SAVE INPUT PARAMETERS -----
2060   ! -----------------------------------------------------------------------
2065   Objsav=Obj
2070   FOR I=1 TO Ndv
2075   Xsav(I)=Xscal(I)
2080   NEXT I
2085   IF Ncon<=0 THEN GOTO A1
2090   FOR I=1 TO Ncon
2095   Tmp(I)=G(I)
2100   NEXT I
2105 A1:  ! CONTINUE
2110   ! -----------------------------------------------------------------------
2115   !                         ----- INITIAL BOUNDS -----
2120   ! -----------------------------------------------------------------------
2125   Alpz=0
2130   Fz=Obj
2135 A6:   ! TRANSFER
2140   FOR I=1 TO Ndv
2145   Xscal(I)=Xsav(I)+Alp3*S(I)
2150   X(I)=Xscal(I)*Scal(I)
2155   NEXT I
2160   !
2165   CALL Analiz(X(*),G(*),Gg(*),C(*),Obj)
2170   Ncal=Ncal+1
2175   IF Ncon>0 THEN GOSUB Penalize
2180   F3=Obj
```

```
2185   IF ABS(Iprint)>4 THEN PRINT "INITIAL BOUNDS."
2190   IF ABS(Iprint)>4 THEN PRINT "ALPHAZ=";Alpz,"FZ=";Fz
2195   IF ABS(Iprint)>4 THEN PRINT "ALPHA3=";Alp3,"F3=";F3
2200   !
2205   ! IF F3 IS GREATER THEN FZ THE MIN. EXISTS BETWEEN ALPZ AND ALP3
2210   IF F3>Fz THEN GOTO A3
2215   ! ----------------------------------------------------------------------
2220 A2:  !          ----- INCREASE THE BOUNDS ON ALPHA -----
2225   ! ----------------------------------------------------------------------
2230   D1=Alp3-Alpz
2235   Alp1=Alp3
2240   F1=F3
2245   Alp3=Alpz+D1/.3819660113
2250   FOR I=1 TO Ndv                          ! INCREMENT X(I) BY ALP3
2255   Xscal(I)=Xsav(I)+Alp3*S(I)
2260   X(I)=Xscal(I)*Scal(I)
2265   NEXT I
2270   !
2275   CALL Analiz(X(*),G(*),Gg(*),C(*),Obj)
2280   Ncal=Ncal+1
2285   IF Ncon>0 THEN GOSUB Penalize
2290   F3=Obj
2295   IF ABS(Iprint)>4 THEN PRINT "INCREASE THE BOUNDS ON ALPHA"
2300   IF ABS(Iprint)>4 THEN PRINT "ALPHAZ=";Alpz,"FZ=";Fz
2305   IF ABS(Iprint)>4 THEN PRINT "ALPHA1=";Alp1,"F1=";F1
2310   IF ABS(Iprint)>4 THEN PRINT "ALPHA3=";Alp3,"F3=";F3
2315   ! ----------------------------------------------------------------------
2320   ! IF OBJ IS GREATER THAN F1
2325   ! THE MINIMUM EXISTS BETWEEN ALPZ AND ALP3.
2330   ! PROCEED TO LOCALIZE THE MINIMUM.
2335   ! OTHERWISE INCREASE THE UPPER AND LOWER BOUNDS
2340   ! AND PROCEED TO SEARCH FOR THE MINIMUM.
2345   ! ----------------------------------------------------------------------
2350   IF F3>F1 THEN GOTO A4
2355   Fz=F1
2360   Alpz=Alp1
2365   GOTO A2
2370   !
```

```
2375 A3: ! CALCULATE ALPHA1 AND F1
2380   D1=Alp3-Alpz
2385   Alp1=Alpz+D1*.3819660113
2390   FOR I=1 TO Ndv
2395   Xscal(I)=Xsav(I)+Alp1*S(I)
2400   X(I)=Xscal(I)*Scal(I)
2405   NEXT I
2410   CALL Analiz(X(*),G(*),Gg(*),C(*),Obj)
2415   Ncal=Ncal+1
2420   IF Ncon>0 THEN GOSUB Penalize
2425   F1=Obj
2430   IF F1>Fz THEN Alp3=Alp1          ! RESTART BRACKETING THE MIN
2435   IF F1>Fz THEN GOTO A6
2440   !
2445 A4: ! CONTINUE
2450   ! ----------------------------------------------------------------------
2455   !                    ----- THE MINIMUM IS BRACKETED -----
2460   ! ----------------------------------------------------------------------
2465   IF ABS(Iprint)=5 THEN PRINT "THE MINIMUM IS BRACKETED - PROCEED TO LOCALI
ZE IT."
2470   D1=Alp3-Alpz
2475   Alp2=Alpz+.6180339887*D1
2480   ! CALCULATE F2
2485   FOR I=1 TO Ndv
2490   Xscal(I)=Xsav(I)+Alp2*S(I)
2495   X(I)=Xscal(I)*Scal(I)
2500   NEXT I
2505   CALL Analiz(X(*),G(*),Gg(*),C(*),Obj)
2510   IF Ncon>0 THEN GOSUB Penalize
2515   Ncal=Ncal+1
2520   F2=Obj
2525 A5: ! TEST FOR LOCALIZATION.
2530   IF ABS(Iprint)>4 THEN PRINT "LOACALIZE THE MINIMUM"
2535   IF ABS(Iprint)>4 THEN PRINT "ALPHAZ";Alpz,"FZ=";Fz
2540   IF ABS(Iprint)>4 THEN PRINT "ALPHA1=";Alp1,"F1=";F1
2545   IF ABS(Iprint)>4 THEN PRINT "ALPHA2=";Alp2,"F2=";F2
2550   IF ABS(Iprint)>4 THEN PRINT "ALPHA3=";Alp3,"F3=";F3
```

```
2555   ! ----------------------------------------------------------------------
2560   ! ----- TEST THE VERTICAL DIFFERENCE ON THE FOUR F's -----
2565   ! ----------------------------------------------------------------------
2570   IF (ABS((Fz-F1)/F1)>.1) OR (ABS((Fz-F2)/F2)>.1) THEN GOTO A11
2575   IF (ABS((F3-F1)/F1)>.1) OR (ABS((F3-F2)/F2)>.1) THEN GOTO A11
2580   ! ----------------------------------------------------------------------
2585   ! TEST TO SEE IF THE CUBIC INTERPOLATOR WILL GIVE A GOOD
2590   ! APPROXIMATION FOR THE MINIMUM ALPHA.
2595   ! ----------------------------------------------------------------------
2600   GOSUB Quebic
2605   Objq=A0+A1*Alpha+A2*Alpha^2+A3*Alpha^3
2610   ! CALCULATE OBJ AT THE ALPHA GIVEN
2615   FOR I=1 TO Ndv
2620   Xscal(I)=Xsav(I)+Alpha*S(I)
2625   X(I)=Xscal(I)*Scal(I)
2630   NEXT I
2635   CALL Analiz(X(*),G(*),Gg(*),C(*),Obj)
2640   IF Ncon>0 THEN GOSUB Penalize
2645   Ncal=Ncal+1
2650 A10:  ! TEST THE TWO RESULTS FOR AGREEMENT
2655   IF ABS(Iprint)>2 THEN PRINT "OBJ=";Obj;"      OBJQ=";Objq
2660   IF ABS((Objq-Obj)/Obj)<1E-3 THEN GOTO A7
2665   IF D1<1E-3 THEN GOTO A7
2670   ! ----------------------------------------------------------------------
2675 A11:  !  ----- PROCEDE WITH THE GOLDEN SEARCH FOR THE MINIMUM -----
2680   ! ----------------------------------------------------------------------
2685   IF D1<1E-12 THEN GOTO A7          ! EVEN THOUGH MIN NOT FOUND - EXIT
2690   IF F1>F2 THEN GOTO A12
2695   IF ABS(Iprint)=5 THEN PRINT "F1<F2"
2700   Alp3=Alp2
2705   F3=F2
2710   Alp2=Alp1
2715   F2=F1
2720   D1=Alp3-Alpz
2725   Alp1=Alpz+.3819660113*D1
```

```
2730   ! CALCULATE F1
2735   FOR I=1 TO Ndv
2740   Xscal(I)=Xsav(I)+Alp1*S(I)
2745   X(I)=Xscal(I)*Scal(I)
2750   NEXT I
2755   CALL Analiz(X(*),G(*),Gg(*),C(*),Obj)
2760   IF Ncon>0 THEN GOSUB Penalize
2765   Ncal=Ncal+1
2770   F1=Obj
2775   GOTO A5
2780   !
2785 A12: ! CONTINUE
2790   IF ABS(Iprint)=5 THEN PRINT "F1 >= F2"
2795   Alpz=Alp1
2800   Fz=F1
2805   Alp1=Alp2
2810   F1=F2
2815   D1=Alp3-Alpz
2820   Alp2=Alpz+.6180339887*D1
2825   ! CALCULATE F2
2830   FOR I=1 TO Ndv
2835   Xscal(I)=Xsav(I)+Alp2*S(I)
2840   X(I)=Xscal(I)*Scal(I)
2845   NEXT I
2850   CALL Analiz(X(*),G(*),Gg(*),C(*),Obj)
2855   IF Ncon>0 THEN GOSUB Penalize
2860   Ncal=Ncal+1
2865   F2=Obj
2870   GOTO A5
2875   !
2880 A7: ! CONTINUE
2885   !
2890   RETURN
2895   !
2900   !
```

```
2905   ! ************************************************************************
2910 Quebic: !              ----- SUBROUTINE QUEBIC -----
2915   ! ************************************************************************
2920   !
2925   !
2930   !   ROUTINE TO ESTIMATE THE ALPHA AT WHICH OBJ IS A MINIMUM BASED
2935   !   ON FOUR POINT CUBIC INTERPOLATION.
2940   ! ------------------------------------------------------------------------
2945   IF ABS(Iprint)<5 THEN GOTO 2975
2950   PRINT "CUBIC INTERPOLATOR INPUT"
2955   PRINT "ALPHAZ=";Alpz,"FZ=";Fz
2960   PRINT "ALPHA1=";Alp1,"F1=";F1
2965   PRINT "ALPHA2=";Alp2,"F2=";F2
2970   PRINT "ALPHA3=";Alp3,"F3=";F3
2975   Q1=Alpz^3*(Alp2-Alp1)-Alp1^3*(Alp2-Alpz)+Alp2^3*(Alp1-Alpz)
2980   Q2=Alpz^3*(Alp3-Alp1)-Alp1^3*(Alp3-Alpz)+Alp3^3*(Alp1-Alpz)
2985   Q3=(Alp2-Alp1)*(Alp1-Alpz)*(Alp2-Alpz)
2990   Q4=(Alp3-Alp1)*(Alp1-Alpz)*(Alp3-Alpz)
2995   Denq=Q2*Q3-Q1*Q4
3000   Q5=F2*(Alp1-Alpz)-F1*(Alp2-Alpz)+Fz*(Alp2-Alp1)
3005   Q6=F3*(Alp1-Alpz)-F1*(Alp3-Alpz)+Fz*(Alp3-Alp1)
3010   A3=(Q3*Q6-Q4*Q5)/Denq
3015   A2=(Q5-Q1*A3)/Q3
3020   A1=(F1-Fz-A3*(Alp1^3-Alpz^3))/(Alp1-Alpz)-A2*(Alpz+Alp1)
3025   A0=Fz-A1*Alpz-A2*Alpz^2-A3*Alpz^3
3030   IF A3=0 THEN GOTO Q10
3035   ! ------------------------------------------------------------------------
3040   ! THE GENERAL EQUATIONS OF THE FUNCTION ARE THEN
3045   !       Y = A0 + A1*X + A2*X^2 + A3*X^3
3050   !   dY/dX = A1 + 2*A2*X + 3*A3*X^2
3055   ! d2Y/dX2 = 2*A2 + 6*A3*X
3060   ! FIND THE MINIMUM USING FIRST AND SECOND DERIVATIVES.
3065   ! ------------------------------------------------------------------------
```

```
3070   ON ERROR GOTO 3080
3075   Xx1=(-A2+(A2^2-3*A1*A3)^.5)/(3*A3)
3080   OFF ERROR
3085   ON ERROR GOTO 3095
3090   Xx2=(-A2-(A2^2-3*A1*A3)^.5)/(3*A3)
3095   OFF ERROR
3100   Y2p=2*A2+6*A3*Xx1
3105   IF Y2p>0 THEN Alpha=Xx1
3110   IF Y2p<0 THEN Alpha=Xx2
3115   IF (Alpha>Alpz) AND (Alpha<Alp3) THEN GOTO Q1
3120   ! ------------------------------------------------------------------------
3125   !    ----- THE PRIDICTED MINIMUM IS OUTSIDE THE BRACKETED MINIMUM -----
3130   ! ------------------------------------------------------------------------
3135   IF F1>F2 THEN GOTO Q2
3140   ! F1<F2  :  FIND THE MINIMUM USING A QUADRATIC FIT
3145   W1=(Alp1-Alpz)*(F2-Fz)-(Alp2-Alpz)*(F1-Fz)
3150   W2=(Alp1-Alpz)*(Alp2-Alpz)*(Alp2-Alp1)
3155   W3=W1/W2
3160   W4=(F1-Fz)/(Alp1-Alpz)-(Alpz+Alp1)*W3
3165   Alpha=-W4/(2*W3)
3170   GOTO Q3
3175 Q2: ! F1 > F2  :  FIND THE MINIMUM USING A QUADRATIC FIT
3180   W1=(Alp2-Alp1)*(F3-F1)-(Alp3-Alp1)*(F2-F1)
3185   W2=(Alp2-Alp1)*(Alp3-Alp1)*(Alp3-Alp2)
3190   W3=W1/W2
3195   W4=(F2-F1)/(Alp2-Alp1)-(Alp1+Alp2)*W3
3200   Alpha=-W4/(2*W3)
3205   GOTO Q3
```

```
3210 Q1:  IF ABS(Iprint)<5 THEN GOTO 3260
3215   PRINT "CUBIC OUTPUT"
3220   PRINT "Q1=";Q1,"Q2=";Q2
3225   PRINT "Q3=";Q3,"Q4=";Q4
3230   PRINT "Q5=";Q5,"Q6=";Q6
3235   PRINT "A0=";A0,"A1=";A1
3240   PRINT "A2=";A2,"A3=";A3
3245   PRINT "XX1=";Xx1,"XX2=";Xx2
3250   PRINT "Y2P=";Y2p
3255 Q3:  IF ABS(Iprint)>2 THEN PRINT "ALPHA=";Alpha
3260   RETURN
3265 Q10:  ! IF A3=0 THEN THE FUNCTION IS A QUADRATIC
3270   Alpha=-A1/(2*A2)
3275   IF ABS(Iprint)<5 THEN GOTO 3295
3280   PRINT "THE FUNCTION IS A QUADRATIC"
3285   PRINT "A1=";A1,"A2=";A2
3290   PRINT "ALPHA=";Alpha
3295   RETURN
3300   !
```

```
3305   ! ************************************************************************
3310 Convrg:  !      ----- SUBROUTINE CONVRG -----
3315   ! ************************************************************************
3320   !
3325   ! ----------------------------------------------------------------------
3330   !                    CHECK ON CONVERGENCE CRITERIA
3335   ! ----------------------------------------------------------------------
3340   !
3345   Del=ABS((Obj-Objsav)/Obj)
3350   Abobj1=(Abobj1+Del)/2
3355   IF Del<Delfun THEN Coni=Coni+1          ! INCREASE CONVERGENCE COUNTER BY 1
3360   IF Del<Delfun THEN Kount=0              ! NEXT ITERATION - STEEPEST DECENT
3365   IF Del<Delfun THEN RETURN
3370   IF ABS(Obj-Objsav)<Dabfun THEN Coni=Coni+1
3375   IF ABS(Obj-Objsav)<Dabfun THEN Kount=0
3380   Objsav=Obj
3385   RETURN
3390   !
3395   !
```

```
3400   ! *********************************************************************
3405 Penalize:!        ----- SUBROUTINE PENALIZE -----
3410   ! *********************************************************************
*
3415   Obja=Obj
3420   ! --------------------------------------------------------------------
3425   !          ----- ADD PENALTIES FOR VIOLATED CONSTRAINTS. -----
3430   ! --------------------------------------------------------------------
3435   FOR I=1 TO Ncon
3440   Gi=G(I)
3445   IF Gi>0 THEN Obj=Obj+Rz*Gi^Expg
3450   NEXT I
3455   ! --------------------------------------------------------------------
3460 Penal1: ! -----  ADD PENALTIES FOR VIOLATED SIDE CONSTRAINTS. -----
3465   ! --------------------------------------------------------------------
3470   FOR I=1 TO Ndv
3475   X(I)=Xscal(I)*Scal(I)
3480   IF X(I)<Vlb(I) THEN Obj=Obj+Rz*((Vlb(I)-X(I))/X(I))^Expg
3485   IF X(I)>Vub(I) THEN Obj=Obj+Rz*((X(I)-Vub(I))/Vub(I))^Expg
3490   NEXT I
3495   RETURN
3500   !
3505   ! -------------------------------------------------------------------
3510   ! -------------------------------------------------------------------
3515   !
3520   !
3525 Analiz1:  ! ATTACHMENT POINT FOR ANALIZE ROUTINE.
3530   !
3535   END
```

APPENDIX F

DESOP Test Programs

```
100  SUB Analiz(X(*),G(*),Gg(*),C(*),Obj)
110   !
120   ! ----- BANNA FUNCTION -----
130   !
140   ! A TEST PROGRAM FOR OPTIMIZER DEVELOPEMENT
150   ! THERE ARE NO CONSTRAINTS ON THE BANANA FUNCTION.
160   ! THERE ARE NO SIDE CONSTRAINTS  ON THE BANANA FUNCTION.
170   !      NDV = 2
180   !      NCON = 0
190   !      RECOMMENDED STARTING VALUES ARE:
200   !         X(1) = -1.2
210   !         X(2) = 1.0
220   !      TRUE MINIMUM = 4.00
230   !
240   ! ----------------------------------------------------------------------
250   !
260   ! DESIGN VARIABLES :
270   X1=X(1)
280   X2=X(2)
290   ! OBJECTIVE FUNCTION :
300   Obj=10*(X2-X1^2)^2+(1-X1)^2+4
310   SUBEND
```

```
100   SUB Analiz(X(*),G(*),Gg(*),C(*),Obj)
110     !
120     ! ----------              ROSEN-SUZUKI FUNCTION                  ----------
130     !                   A CONSTRAINED OPTIMIZATION PROBLEM
140     !
150     ! A TEST PROGRAM FOR OPTIMIZER DEVELOPEMENT
160     ! THERE ARE NO SIDE CONSTRAINTS ON THE ROSEN-SUZUKI FUNCTION.
170     !    NDV = 4
180     !    NCON = 3
190     !    RECOMMENDED STARTING VALUES:
200     !       ALL X(I) = 1
210     !    TRUE MINIMUM = 6.00
220     !
230     ! ------------------------------------------------------------------------
240     !
250     ! DESIGN VARIABLES :
260     X1=X(1)
270     X2=X(2)
280     X3=X(3)
290     X4=X(4)
300     ! OBJECTIVE FUNCTION :
310     Obj=X1^2-5*X1+X2^2-5*X2+2*X3^2-21*X3+X4^2+7*X4+50
320     ! CONSTRAINTS
330     G(1)=X1^2+X1+X2^2-X2+X3^2+X3+X4^2-X4-8
340     G(2)=X1^2-X1+2*X2^2+X3^2+2*X4^2-X4-10
350     G(3)=2*X1^2+2*X1+X2^2-X2+X3^2-X4-5
360     !
370     SUBEND
```

```
100   SUB Analiz(X(*),G(*),Gg(*),C(*),Obj)
110   GOTO 490
120   !
130   !                         "TSVAR"
140   !
150   !                A CONSTRAINED OPTIMIZATION PROBLEM
160   !
170   !    Himmelblau, D. M. , Applied Nonlinear Programming, McGraw Hill Book
180   !         Co., San Francisco, 1972, pp.410-412
190   !
200   !    CODED BY : Walter B. Cole, June 9, 1980
210   !
220   !      NDV = 5
230   !      NCON = 6
240   !      THERE ARE 10 SIDE CONSTRAINTS
250   !      RECOMMENDED STARTING VALUES ARE :
260   !         X(1) = 2.52
270   !         X(2) = 2
280   !         X(3) = 37.5
290   !         X(4) = 9.25
300   !         X(5) = 6.8
310   !      SIDE CONSTRIAINTS TO BE ENTERED ON INPUT
320   !         1.      0 <= X1 <= 5
330   !         2.    1.2 <= X2 <= 2.4
340   !         3.     20 <= X3 <= 60
350   !         4.      9 <= X4 <= 9.3
360   !         5.    6.5 <= X5 <= 7
370   !
380   !      FINAL RESULTS :
390   !         OBJ  = -5280254
400   !         X(1) = 4.538
410   !         X(2) = 2.400
420   !         X(3) = 60.000
430   !         X(4) = 9.300
440   !         X(5) = 7.000
450   !
```

```
460   ! --------------------------------------------------------------------------
470   !
480   ! DESIGN VARIABLES :
490   X1=X(1)
500   X2=X(2)
510   X3=X(3)
520   X4=X(4)
530   X5=X(5)
540   !
550   ! CONSTANTS :
560   K1=-145421.402
570   K2=2931.1506
580   K3=-40.427932
590   K4=5106.192
600   K5=15711.36
610   K6=-161622.577
620   K7=4176.15328
630   K8=2.8260078
640   K9=9200.476
650   K10=13160.295
660   K11=-21686.9194
670   K12=123.56928
680   K13=-21.1188894
690   K14=706.834
700   K15=2898.573
710   K16=28298.388
720   K17=60.81096
730   K18=31.242116
740   K19=329.574
750   K20=-2882.082
760   K21=74095.3845
770   K22=-306.262544
780   K23=16.243649
790   K24=-3094.252
800   K25=-5566.2628
810   K26=-26237
820   K27=99
```

```
830   K28=-.42
840   K29=1300
850   K30=2100
860   K31=925548.252
870   K32=-61968.8432
880   K33=23.3088196
890   K34=-27097.648
900   K35=-50843.766
910   !
920   X6=(K1+K2*X2+K3*X3+K4*X4+K5*X5)*X1
930   Y1=K6+K1*X2+K8*X3+K9*X4+K10*X5
940   Y2=K11+K12*X2+K13*X3+K14*X4+K15*X5
950   Y3=K16+K17*X2+K18*X3+K19*X4+K20*X5
960   Y4=K21+K22*X2+K23*X3+K24*X4+K25*X5
970   X7=(Y1+Y2+Y3)*X1
980   X8=(K26+K27*X2+K28*X3+K29*X4+K30*X5)*X1+X6+X7
981   Icalc=C(1)
990   IF Icalc>1 THEN GOTO 1080
1000  PRINTER IS 0
1010  PRINT "                TSVAR INTERNAL PARAMETERS"
1020  PRINT LIN(2),"X6=";X6,"Y1=";Y1
1030  PRINT "Y2=";Y2,"Y3=";Y3
1040  PRINT "Y4=";Y4,"X7=";X7
1050  PRINT "X8=";X8
1060  PRINTER IS 16
1070  !
1080  ! OBJECTIVE FUNCTION :
1090  Max=(50*Y1+9.583*Y2+20*Y3+15*Y4-852960-38100*(X2+.01*X3)+K31+K32*X2+K33*X3
+K34*X4+K35*X5)*X1-24345+15*X6
1100  Obj=-Max
```

```
1110  !
1120  ! CONSTRAINTS :
1130  G(1)=-X6
1140  G(2)=X6-294000
1150  G(3)=-X7
1160  G(4)=X7-294000
1170  G(5)=-X8
1180  G(6)=X8-277200
1190  !
1200  !
1210  SUBEND
```

```
100   SUB Analiz(X(*),G(*),Gg(*),C(*),Obj)
110   !
120   !                         "T7VAR"
130   !
140   !              A CONSTRAINED OPTIMIZATION PROBLEM
150   !
160   !  Kuester, J. L., and Mize, J. H.,  Optimization Techniques,
170   !       McGraw Hill, San Francisco 1973, pp.73-74.
180   !
190   ! A TEST PROGRAM FOR OPTIMIZER DEVELOPEMENT.
200   !     NDV = 7
210   !     NCON = 1
220   !     SIDE CONSTRAINTS: ALL X(I) > 0
230   !     TRUE MINIMUM = -200
240   !
250   ! ------------------------------------------------------------------------
260   !
270   ! NOTE : SIDE CONSTRAINTS ENTERED ON INPUT, ALL X(I) > 0
280   !
290   ! DESIGN VARIABLES :
300   X1=X(1)
310   X2=X(2)
320   X3=X(3)
330   X4=X(4)
340   X5=X(5)
350   X6=X(6)
360   X7=X(7)
370   ! OBJECTIVE FUNCTION :
380   Obj=-60*X1-60*X2-40*X3-10*X4-20*X5-10*X6-3*X7
390   ! CONSTRAINTS :
400   G(1)=-10+3*X1+5*X2+4*X3+1*X4+4*X5+3*X6+1*X7
410   ! SIDE CONSTRAINTS : TO BE ENTERED ON INPUT
420   !     ALL X(I) >= 0
430   !
440   SUBEND
```

APPENDIX G

NISCO Subprogram Listing

Below is a cross reference list of major equations used in the NISCO program to the references used to develop the equations.

Reference Number	NISCO Subprogram Line Number
3.	820 - 850
4.	1085 - 1090
5.	915, 985 - 1045
6.	855, 925, 930, 965 - 980, 1100, 1180 - 1210, 1255 - 1345, 1460, 1465,
11.	905, 1115, 1125, 1145
12.	1445 - 1455, 1470 - 1480
13.	1600 - 1605

```
100  SUB Analiz(X(*),G(*),Gg(*),C(*),Obj)
105   !
110   !    ******************************************************************
115   !    *                                                                *
120   !    *                           NISCO                                *
125   !    *                                                                *
130   !    *           AN OPTIMIZATION ANALYSIS PROGRAM FOR A               *
135   !    *           NONIMAGING COMPOUND PARABOLIC TROUGH                 *
140   !    *           CONCENTRATING SOLAR COLLECTOR.                       *
145   !    *                                                                *
150   !    *           by WALTER B. COLE                                    *
155   !    *           NAVAL POST GRADUATE SCHOOL, MONTEREY, CA.            *
160   !    *           1980                                                 *
165   !    *                                                                *
170   !    ******************************************************************
175   !
180   DIM Ds(13),Nhid(13),Ib(13,10),Alts(13,10),Azm(13,10),Tamb(13),Drf(13)
185   !
190   Icalc=C(1)                        ! OPTIMIZER FLAG FOR 1st AND LAST RUN
195   Pi=3.141592654
200   Ncal=Ncal+1                       ! ITERATION COUNTER
205   Thetai=X(1)*PI/180                ! MAXIMUM ACCEPTANCE ANGLE (Radoans)
210   Thetat=X(2)*PI/180                ! TRUNCATED THETA (Radians)
215   R=X(3)/12                         ! RECEIVER RADIUS (Fteet)
220   L=X(4)                            ! RECEIVER LENGTH (Feet)
225   Mfr=X(5)                           ! MASS FLOW RATE (lbm/Hour)
230   !
235   ! -------------------------------------------------------------------
240 N1:  !              ----- CONSTANTS -----
245   ! -------------------------------------------------------------------
250   Lat=40*PI/180                      ! 40 DEGREES NORTH LATITUDE
255   Awall=0                            ! WALL AZIMUTH ANGLE w/r TO SOUTH
260   Alpr=.93                           ! RECEIVER ABSORPTIVITY (OXIDIZED Cu)
265   Eptr=.40                           ! RECEIVER EMISSIVITY (THERMAL RAD.)
270   Rhor=.07                           ! RECEIVER REFLECTIVITY
275   Rhom=.89                           ! REFLECTOR REFLECTIVITY (VACUUM
280                                      !   DEPOSITED Al ON RESIN)
```

```
285   Alpab=.03                              ! COVER AVERAGE ABSOPTANCE
290   Taub=.8                                ! COVER AVERAGE TRANSMITTANCE
295   Rhog=.17                               ! COVER AVERAGE REFLECTIVITY
300   Sbc=1.714E-9                           ! STEPHAN BOLTZMAN CONSTANT
305                                          ! ( BTU/(Hr Ft^2 R^4) )
310   Gc=32.2                                ! GRAVITY CONSTANT
315   Epta=.94                               ! COVER EMISSIVITY (THERMAL RAD.)
320   Vp=3                                   ! VAPOR PRESSURE (mm Hg)
325   A3=.68                                 ! LATTITUDE FUNCTION
330   Cc=0                                   ! CLOUD COVER (TENTHS)
335   Rhab=.07                               ! COVER AVERAGE REFLECTIVITY
340   Tc1=100                                ! COOLANT INLET TEMP TO RECIEVER (F)
345   Sg=.87                                 ! COOLANT SPECIFIC GRAVITY
350   Zzz=1                                  ! ITERATION COUNTER
355   Iprint=7                               ! PRINTER CONTROL
360   Dt=.0001                               ! CONVERGENCE CRITERIA (fdchm/100)
365   Nyrs=20                                ! NUMBER OF YEARS
370   Cac=22.5                               ! COLLECTOR COST ( $/ft^2 )
375   Cf=8.14E-6                             ! FUEL COST ( $/Btu ) FOR GAS
380   Ia=.1                                  ! ANNUAL INFLATION RATE
385   If=.11                                 ! ANNUAL FUEL INFLATION RATE
390   Dtcm=10                                ! MINIMUM CHANGE IN COOLANT TEMP
395   !
400   ! ------------------------------------------------------------------------
405 N2:   !                      ----- DATA -----
410   ! ------------------------------------------------------------------------
415   ! Ds(M) - SOLAR DECLINATION (MONTH)
420   DATA -0.34907,-.188496,0,.20246,0.349066,0.409279,.35954,0.214675,0,-.1832
6,-.345575,-.409279
425   ! Nhid(M) - NUMBER OF SOLAR INSOLATION HOUR PERIODS / DAY (MONTH)
430   DATA 5,6,6,7,8,8,8,7,6,6,5,5
435   ! Ib(M,H) - TOTAL INSOLATION ON NORMAL SURFACE (MONTH,HOUR) (BTU/Ft^2)
```

```
436   ! DATA FOR 40 deg NORTH LATTITUDE FROM ASHRAE HANDBOOK OF FUNDAMENTALS
440   DATA 294,289,274,239,142                        ! JAN. 21
445   DATA 308,305,295,274,224,69                     ! FEB. 21
450   DATA 307,305,297,282,250,171                    ! MAR. 21
455   DATA 293,292,286,274,252,206,89                 ! APR. 21
460   DATA 284,283,277,267,250,216,144,1              ! MAY  21
465   DATA 279,277,272,263,246,216,155,22             ! JUN. 21
470   DATA 276,275,269,259,241,208,138,2              ! JUL. 21
475   DATA 280,278,272,260,237,191,81                 ! AUG. 21
480   DATA 290,287,280,263,230,149                    ! SEP. 21
485   DATA 294,291,280,257,204,48                     ! OCT. 21
490   DATA 288,283,268,232,136                        ! NOV. 21
495   DATA 285,280,261,217,89                         ! DEC. 21
500   ! Alts(M,H) - SOLAR ALTITUDE (MONTH,HOUR)
505   DATA 30,28.4,23.8,16.8,8.1                      ! JAN. 21
510   DATA 40,38.1,32.8,25,15.4,4.8                   ! FEB. 21
515   DATA 50,47.7,41.6,32.8,22.5,11.4                ! MAR. 21
520   DATA 61.6,58.7,51.2,41.3,30.3,18.9,7.4          ! APR. 21
525   DATA 70,66.2,57.5,46.8,35.4,24.0,12.7,1.9       ! MAY 21
530   DATA 73.5,69.2,59.8,48.8,37.4,26.0,14.8,4.2     ! JUN. 21
535   DATA 70.6,66.7,57.9,47.2,35.8,24.3,13.1,2.3     ! JUL. 21
540   DATA 62.3,59.3,51.7,41.8,30.7,19.3,7.9          ! AUG. 21
545   DATA 50.0,47.7,41.6,32.8,22.5,11.4              ! SEP. 21
550   DATA 39.5,37.6,32.4,24.5,15.0,4.5               ! OCT. 21
555   DATA 30.2,28.6,24.0,17.0,8.2                    ! NOV. 21
560   DATA 26.6,25.0,20.7,14.0,5.5                    ! DEC. 21
565   ! Azm(M,H) - SOLAR AZIMUTH ANGLE (MONTH,HOUR)
570   DATA 0,16,30.9,44,55.3                          ! JAN. 21
575   DATA 0,18.9,35.9,50.2,62.2,72.7                 ! FEB. 21
580   DATA 0,22.6,41.9,57.3,69.6,80.2                 ! MAR. 21
585   DATA 0,29.2,51.4,67.2,79.3,89.5,98.9            ! APR. 21
590   DATA 0,37.1,60.9,76.0,87.2,96.6,105.6,114.7     ! MAY  21
595   DATA 0,41.9,65.8,80.2,90.7,99.7,108.4,117.3     ! JUN. 21
600   DATA 0,37.9,61.7,76.7,87.8,97.2,106.1,115.2     ! JUL. 21
605   DATA 0,29.7,52.1,67.9,79.9,90.9,99.5            ! AUG. 21
610   DATA 0,22.6,41.9,57.3,69.6,80.2                 ! SEP. 21
615   DATA 0,18.7,35.6,49.8,61.9,72.3                 ! OCT. 21
620   DATA 0,16.1,31.0,44.1,55.4                      ! NOV. 21
625   DATA 0,15.2,29.4,41.9,53                        ! DEC. 21
```

```
630   ! Tamb(M) - AVERAGE DAILY TEMPERATURE (MONTH)
635   DATA 40,40,45,55,60,70,80,80,70,60,45,40
640   ! Drf(M) - DIFFUSE SOLAR RADIATION FACTOR (MONTH)
645   DATA 0.058,0.06,0.071,.097,.121,.134,.136,.122,.092,.073,.063,.057
650   ! READ INPUT DATA
655   FOR M=1 TO 12
660   READ Ds(M)
665   NEXT M
670   FOR M=1 TO 12
675   READ Nhid(M)
680   NEXT M
685   FOR M=1 TO 12
690   FOR H=1 TO Nhid(M)
695   READ Ib(M,H)
700   NEXT H
705   NEXT M
710   FOR M=1 TO 12
715   FOR H=1 TO Nhid(M)
720   READ Altsd
725   Alts(M,H)=Altsd*PI/180                              ! CHANGE TO RADIANS
730   NEXT H
735   NEXT M
740   FOR M=1 TO 12
745   FOR H=1 TO Nhid(M)
750   READ Azmd
755   Azm(M,H)=Azmd*PI/180
760   NEXT H
765   NEXT M
770   FOR M=1 TO 12
775   READ Tamb(M)
780   NEXT M
785   FOR M=1 TO 12
790   READ Drf(M)
795   NEXT M
800   !
```

```
805   ! ------------------------------------------------------------------------------
810 N3:  !               ----- COLLECTOR GEOMETRY -----
815   ! ------------------------------------------------------------------------------
820   T=R*(Thetat+Thetai+PI/2-COS(Thetat-Thetai))/(1+SIN(Thetat-Thetai))
825   Thetax=3*PI/2-Thetai                          ! THETA MAX.
830   Thetan=PI/2+Thetai                            ! THETA MIN.
835   At=2*(R*SIN(Thetat)-T*SIN(Thetat+PI/2))! APERATURE OPENING AREA
840   Ar=2*PI*R                                     ! RECEIVER AREA
845   Cr=At/Ar                                      ! CONCENTRATION RATIO
850   Cd=-R*COS(Thetat)+T*COS(Thetat+PI/2)          ! COLLECTOR DEPTH (ft)
855   Nbar=LOG(Cr^.5)                               ! AVERAGE NUMBER OF REFLECTIONS
860   !
865   Tr=150                                        ! ASSUME AN INITIAL RECEIVER TEMP
870   Ta=100                                        ! ASSUME AN INITIAL COVER TEMP
875   !
880   ! ------------------------------------------------------------------------------
885 N4:  !               ----- MONTHLY CALCULATIONS -----
890   ! ------------------------------------------------------------------------------
895   !
900   FOR M=1 TO 12
905   Beta=Thetai-Alts(M,1)+PI/2                 ! COLLECTOR TILT ANGLE
910   Betag=PI/2-Beta-Thetai                     ! MINIMUM ACCEPTED SOLAR ALTITUDE
915   Fag=(1-COS(Beta))/2                        ! GROUND ANGLE FACTOR
920   ! SKY HEAT LOSS CONSTANTS
925   Asky=Epta*Sbc*(Tamb(M)+460)^4*(.39-.0096*Vp)*(1-A3*Cc)-4*Epta*Sbc*(Tamb(M)
+460)^4
930   Bsky=4*Epta*Sbc*(Tamb(M)+460)^3
935   ! ------------------------------------------------------------------------------
940 N5: !               ----- HOURLY CALCULATIONS -----
945   ! ------------------------------------------------------------------------------
950   FOR H=1 TO Nhid(M)
955   Hs=(H-1)*15*PI/180                             ! SOLAR HOUR ANGLE
960   ! Csi - COSINE OF THE ANGLE THE SUN MAKES WITH THE COVER NORMAL
965   Csi1=SIN(Ds(M))*(SIN(Lat)*COS(Beta)-COS(Lat)*SIN(Beta)*COS(Awall))
970   Csi2=COS(Ds(M))*COS(Hs)*(COS(Lat)*COS(Beta)+SIN(Lat)*SIN(Beta)*COS(Awall))
975   Csi3=COS(Ds(M))*SIN(Beta)*SIN(Awall)*SIN(Hs)
980   Csi=Csi1+Csi2+Csi3
985   ! Hpct - PERCENTAGE OF A PARTIAL HOUR OF RADIATION
```

```
990   Hpct=(Alts(M,H-1)-Betag)/(Alts(M,H-1)-Alts(M,H))
995   ! Ibc - DIRECT BEAM RADIATION INCIDENT ON THE COVER
1000  Ibc=Ib(M,H)*Csi
1005  IF Alts(M,H)<Betag THEN Ibc=Ibc*Hpct
1010  ! Idc - DIFFUSE BEAM RADIATION INCIDENT ON THE COVER
1015  Idc=Drf(M)*Ibc*(1-Fsg)
1020  ! Irc - REFLECTED BEAM RADIATION INCIDENT ON THE COVER
1025  Irc=Ibc*(Drf(M)+SIN(Alts(M,H)))*Rhog*Fsg
1030  ! Tauai - COVER TRANSMISIVITY AS A FUNCTION OF INCIDENT ANGLE
1035  Tauai=-.00885+2.71235*Csi-.62062*Csi^2-7.07329*Csi^3+9.75995*Csi^4-3.89922
*Csi^5
1040  ! Alpai - COVER ABSORPTANCE AS A FUNCTION OF INCIDENT ANGLE
1045  Alpai=.01154+.77674*Csi-3.94657*Csi^2+8.57881*Csi^3-8.38135*Csi^4+3.01188*
Csi^5
1050  Zzz=1                                   ! ITERATION COUNTER
1055  ! -----------------------------------------------------------------------
1060 N6:!     ----- PERFORM A HEAT BALANCE ON THE COLLECTOR -----
1065  ! -----------------------------------------------------------------------
-
1070  Zzz=Zzz+1
1075  Trr=Tr+460
1080  Tar=Ta+460
1085  Cp=4.94E-4*Tr+.4036                     ! SPECIFIC HEAT FOR THERMINOL 55
1090  Visc=(-.053*Tr+32.3)*6.71955E-4         ! VISCOSITY (lbm/(ft sec))
1095  ! CALCULATE THE HEAT TRANSFER COEFFICIENTS.
1100  ! Hrar - RADIATION HEAT TRANSFER COEFFICIENT : COVER TO RECEIVER
1105  Hrar=Sbc*(Trr+Tar)*(Trr^2+Tar^2)/(1/Eptr+1/Epta-1)
1110  ! Hcra - CONVECTION HEAT TRANSFER COEFFICIENT : RECEIVER  TO COVER
1115  Hcra=.27*(ABS(Tr-Ta)/(2*Ar))^.25
1120  ! Hcar - CONVECTION HEAT TRANSFER COEFFICIENT : COVER TO RECEIVER
1125  Hcar=.29*(ABS(Ta-Tr)/2*SIN(Beta)/At)^.25
1130  ! Hcb - AVERAGE HEAT TRANSFER COEFFICIENT BETWEEN COVER AND RECEIVER
1135  Hcb=Hcra*Hcar/(Cr*(Hcar+Hcra/Cr))
1140  ! Hce - CONVECTION HEAT TRANSFER COEFFICIENT COVER TO ENVIRONMENT
1145  Hce=.29*(ABS(Ta-Tamb(M))*SIN(Beta)/At)^.25+Cr
```

```
1150  ! ------------------------------------------------------------------------
1155 N7:! ---- PERFORM A HEAT BALANCE ON THE COVER AND SOLVE FOR TA -----
1160  ! ------------------------------------------------------------------------
1165  !
1170  ! Qba - DIRECT SOLAR RADIATION ABSORBED BY THE COVER BOTH DIRECTLY AND
1175  !          INDIRECTLY AFTER REFLECTION FROM THE RECEIVER
1180  Qba=Ibc*(Alpai+Tauai*Rhom^(2*Nbar)*Rhor*Alpab)*Cr
1185  ! Qda - DIFFUSE SOLAR RADIATION ABSORBED BY THE COVER
1190  Qda=Idc*Alpab*Cr
1195  ! Qra - REFLECTED SOLAR RADIATION ABSORBED BY THE COVER
1200  Qra=Irc*Alpab*Cr
1205  ! Tap - NEW COVER TEMPERATURE
1210  Tap=(Qba+Qda+Qra-Asky-Bsky*460+Tr*(Hrar+Hcb)+Hce*Tamb(M))/(Hrar+Hcb+Bsky+H
ce)
1215  IF ABS(Tap>500) THEN Tap=70
1220  ! ------------------------------------------------------------------------
1225 N8:!      ----- PERFORM A HEAT BALANCE ON THE RECEIVER -----
1230  !          ----- AND SOLVE FOR THE USEFUL HEAT OUT -----
1235  ! ------------------------------------------------------------------------
1240  !
1245  ! Qbr - DIRECT SOLAR RADIATION ABSORBED BOTH DIRECTLY AND INDIRECTLY
1250  !          BY THE RECEIVER AFTER REFLECTION FROM THE REFLECTOR
1255  Qbr=Ibc*Tauai*Rhom^Nbar*Alpr*(1+Rhom^(2*Nbar)*Rhor*Rhab)*Cr
1260  ! Qdr - DIFFUSE SOLAR RADIATION ABSORBED BY THE RECEIVER
1265  Qdr=Idc*Taub*Rhom^Nbar*Alpr
1270  ! Qrr - REFLECTED SOLAR RADIATION ABSORBED BY THE RECEIVER
1275  Qrr=Irc*Taub*Rhom^Nbar*Alpr
1280  ! Qcra - CONVECTIVE EXCHANGE BETWEEN THE RECEIVER AND THE COVER
1285  Qcra=Hcb*(Tr-Ta)
1290  ! Qir - RADIATIVE EXCHANGE BETWEEN THE RECEIVER AND THE COVER
1295  Qir=Hrar*(Tr-Ta)
1300  ! Qu - USEFUL HEAT EXTRACTION
1305  Qu=(Qbr+Qdr+Qrr-Qcra-Qir)*L*Ar
1310  !
```

```
1315  ! ------------------------------------------------------------------------
1320 N9:!    ----- PERFORM A HEAT BALANCE ON THE WORKING FLUID -----
1325  !           ----- AND SOLVE FOR THE NEW RECEIVER TEMP -----
1330  ! ------------------------------------------------------------------------
1335  !
1340  ! Tc2 - COOLANT EXIT TEMPERATURE
1345  Tc2=Qu/(Cp*Mfr)+Tc1
1350  IF (M=6) AND (H=1) THEN Tc2i=Tc2
1355  ! Trp - NEW RECIEVER TEMPERATURE
1360  Trp=(Tc1+Tc2)/2
1365  IF ABS(Trp)>10000 THEN Trp=10000
1370  ! ------------------------------------------------------------------------
1375 N10:!   ----- CHECK FOR CONVERGENCE OF Ta WITH Tap AND Tr WITH Trp -----
1380  ! ------------------------------------------------------------------------
1385  !
1390  IF (ABS((Ta-Tap)/Tap)<Dt) AND (ABS((Tr-Trp)/Trp)<Dt) THEN GOTO N11
1395  Ta=Tap
1400  Tr=Trp
1405  GOTO N6
1410  !
1415 N11:  ! CONVERGENCE MET
1420  !
1425  ! ------------------------------------------------------------------------
1430 N12: !     ----- CALCULATE PUMPING POWER REQUIRED -----
1435  ! ------------------------------------------------------------------------
1440  !
1445  Den=Sg*62.4                            ! COOLANT DENSITY (lbm/ft^3)
1450  Mv=Mfr/(PI*R^2)                        ! COOLANT MASS VELOCITY (lbm/hr/ft^2)
1455  Re=Mv*2*R/(Visc*3600)                  ! REYNOLDS NUMBER
1460  IF Re<2100 THEN Ff=16/Re               ! FRICTION FACTOR
1465  IF Re>=2100 THEN Ff=.079*Re^(-.25)
1470  Vc=Mfr/(Den*PI*R^2*3600)               ! COOLANT VELOCITY (ft/sec)
1475  Dpc=2*Ff*Den*Vc^2/(R*Gc)
1480  Ppc=Sg*Mfr*Dpc*1.84939E-2*L/Den        ! COLLECTOR PUMPING POWER (Btu/hr)
1485  IF Alts(M,H)<Betag THEN Ppc=Ppc*Hpct
```

```
1490  Qa=Qu-Ppc                                 ! COLLECTOR ENERGY AVAILABLE
1495  IF (M=6) AND (H=1) THEN Qai=Qa            ! INSTANTANEOUS HEAT GAIN
1500  IF Qa<0 THEN GOTO N14                     ! WHEN PUMP REQ. > Q COLLECTED
1505  IF H>1 THEN Qa=2*Qa                       ! ALLOWS FOR DOUBLE SOLAR PERIOD
1510  ! Qtot - TOTAL USEFUL ENERGY OUT
1515  Qtot=Qtot+Qa
1520  Htot=Htot+(Hhid-1)*2+1                    ! TOTAL NUMBER OF HOURS
1525  IF Alts(M,H)<Betag THEN GOTO N14
1530 N13: ! ---------------------------------------------------------------------
-
1535                              NEXT H
1540  ! ------------------------------------------------------------------------
1545 N14: ! ----------------------------------------------------------------------
1550                              NEXT M
1555  ! ------------------------------------------------------------------------
1560  !
1565  ! ------------------------------------------------------------------------
1570 N15: !  ----- PERFORM AN ECONOMIC ANALYSIS ON THE SYSTEM -----
1575  ! ------------------------------------------------------------------------
1580  !
1585  ! ------------------------------------------------------------------------
1590  Qyr=30*Qtot                              ! YEARLY HEAT GAIN
1595  Ce=Qyr*Cf                                 ! FIRST YEAR FUEL SAVINGS
1600  Ir=Ia-If-Ia*If                            ! EQUIVALENT ANNUAL INTEREST RATE
1605  Clc=Ce*((1+Ir)^Nyrs-1)/(Ir*(1+Ir)^Nyrs)
1610  Ci=Cac*L*(.8*At+.2*Cd)                    ! INITIAL COST ( $/ft^3 ) COLLECTOR
1615  Ctot=Ci-Clc                               ! TOTAL COST (-COST REP. SAVINGS)
1620  IF Qyr=0 THEN Qyr=1E-10
1625  Obj=Ctot/(Qyr*Nyrs)*1E6
1630  !
```

```
1635  ! ----------------------------------------------------------------------------
1640 N16: !                  ----- CONSTRAINTS -----
1645  ! ----------------------------------------------------------------------------
1650  !
1655  Q1=50000
1660  G(1)=Thetai-Thetat-3*PI/2                ! MAXIMUM THETA-T
1665  G(2)=Thetai+PI/2-Thetat                  ! MINIMUM THETA-T
1670  G(3)=-(Mfr-1)*10
1675  G(4)=-(X(3)-.1)*10
1680  G(5)=1-Qtot/(2*Q1)                     ! MINIMUM AVERAGE DAILY HEAT GAIN
1685  G(6)=Tc2i-600                            ! MAXIMUM COLLECTOR TEMPERATURE
1690  IF Icalc=2 THEN GOTO N18
1695  ! ----------------------------------------------------------------------------
1700 N17:  !  .                 ----- OUTPUT -----
1705  ! ----------------------------------------------------------------------------
1710  PRINTER IS 0
1715  IF Icalc=1 THEN PRINT "INITIAL DESIGN :"
1720  IF Icalc=3 THEN PRINT "FINAL DESIGN :"
1725  PRINT "FOR ";Q1;" BTU/DAY SOLAR COLLECTOR"
1730  PRINT "DESIGN VARIABLES :"
1735  PRINT "   INCIDENT ACCEPTANCE ANGLE = ";X(1);" DEGREES"
1740  PRINT "   TRUNCATION ANGLE = ";X(2);" DEGREES"
1745  PRINT "   RECEIVER RADIUS = ";X(3);" INCHES"
1750  PRINT "   COLLECTOR LENGTH = ";X(4);" FEET"
1755  PRINT "   COOLANT MASS FLOW RATE = ";X(5);" Lbm/Hr "
1760  PRINT "DESIGN FEATURES :"
1765  PRINT "   CONCENTRATION RATIO = ";Cr
1770  PRINT "   COLLECTOR APERATURE AREA = ";At*L;" Ft^2"
1775  PRINT "   COLLECTOR DEPTH = ";Cd*12;" INCHES"
1780  PRINT "   COOLANT VELOCITY = ";Vc;" Ft/sec"
1785  PRINT "   MAXIMUM COOLANT TEMPERATURE = ";Tc2i;" DEG F"
1790  PRINT "   INSTATANEOUS COLLECTOR EFFICIENCY = ";Qai/(At*L*279)
1795  PRINT "   AVERAGE DAILY HEAT GAIN = ";Qtot/2;" Btu"
1800  PRINT "   FIRST YEAR FUEL SAVINGS = $";Ce
1805  PRINT "   LIFE CYCLE FUEL SAVINGS = $";Clc
1810  PRINT "   INITIAL COST = $";Ci
1815  PRINT "   TOTAL LIFE CYCLE COST/ 1E6 Btu = $";Obj
1820 N18:  ! END
1825  SUBEND
```

APPENDIX H

Sample DESOP Output

Appendix H is a sample computer output for the DESOP program.

```
ANALIZE PROGRAM :   ROSEN

OPTIMIZER USED :    DESOP

INPUT FOR OPTIMIZATION :

NDV= 4              NCON= 3              IPRINT= 5            DELFUN= .001
DABFUN= .001        ITMAX= 20            ICNDIR= 5            FDCH= .0001
FDCHM= .0001        ABOBJ1= .1           ALPHAX= .1           IPDBG= 0
CT= .1              CTMIN=-.004          CTL= .1              CTLMIN=-.001
PHI= 5              THETA= 1             ITRM= 3              CG= 0
NCF= 0              IRMAX= 10            RZ= 2                RMULT= 2
EXPG= 1.5           NSCAL= 1

X( 1 ) =  1         X( 2 ) =  1          X( 3 ) =  1          X( 4 ) =  1

VLB(1) = -1.00000000000E+50          VUB(1) =  1.00000000000E+50
VLB(2) = -1.00000000000E+50          VUB(2) =  1.00000000000E+50
VLB(3) = -1.00000000000E+50          VUB(3) =  1.00000000000E+50
VLB(4) = -1.00000000000E+50          VUB(4) =  1.00000000000E+50
```

```
--------------------------------------------------------------------------------

                              INITIAL DESIGN

DESIGN VARIABLES :
X(1)= 1           X(2)= 1           X(3)= 1           X(4)= 1

CONSTRAINTS :
G(1)=-4           G(2)=-6           G(3)=-1

OBJECTIVE FUNCTION :
OBJ= 31           OBJA= 31
```

```
------------------------------------------------------------------------

                       OPTIMIZATION  RESULTS

FINAL DESIGN VARIABLES :
X(1) = -5.16678781285E-03                 X(2) =  1.01911857463
X(3) =  1.99950170832                     X(4) = -.995109574088

FINAL CONSTRAINTS :
G(1)=-.00279456917  G(2)=-.9439983936   G(3)= .00232056596

FINAL OBJECTIVE FUNCTION :
OBJ= 6.006979126    OBJA= 5.999824765


NUMBER OF ITERATIONS : 28


NUMBER OF TIMES ANALIZE CALLED :        306


RZ HAS BEEN INCREASED  5  TIMES TO  64
```

LIST OF REFERENCES

1. Kuester, J. L., and Mize, J. H., Optimization Techniques, PP. 73-74, 344-345, McGraw-Hill, 1973.

2. Himmelblau, D. M., Applied Nonlinear Programming, pp. 4-5, 42-44, McGraw-Hill, 1972.

3. Welford, W. T. and Winston, R., The Optics of Nonimaging Concentrators: Light and Solar Energy, pp. 5, 13, 50-52, 83-84, 92, 189-191, Academic Press, 1978.

4. Solar Products Specification Guide, Solar Age Magazine, Solar Vision Inc., 1980.

5. ASHRAE Handbook of Fundamentals, American Society of Heating Refrigeration and Air-Conditioning Engineers, Inc., pp. 386-399, 562-564, 669-681, 1972.

6. Kreith, F. and Kreider, J. F., Principles of Solar Engineering, pp. 59, 90-99, 208, 244, 264, 282-284, 509, 672, McGraw-Hill, 1978.

7. Vanderplaats, G. N., COPES - A FORTRAN Control Program for Engineering Synthesis, pp. 1-73, paper presented at the Naval Postgraduate School during the Engineering Design Optimization Course, 1980.

8. Vanderplaats, G. N., Numerical Optimization Techniques for Engineering Design, pp. 1-21, a paper presented at the Naval Postgraduate School during the Engineering Design Optimization Course, 1980.

9. NASA Technical Paper 1370, Approximation Concepts for Numerical Airfoil Optimization, pp. 2-5, by G. N. Vanderplaats, 1979.

10. Class notes from Engineering Design Optimization, a course given at the Naval Postgraduate School, 1980.

11. Kreider, J. F. and Kreith, F., Solar Heating and Cooling: Engineering Practical Design and Economics, pp. 246, 255, 258, McGraw-Hill, 1975.

12. ASHRAE Handbook of Equipment, American Society of Heating Refrigeration and Air-Conditioning Engineers, Inc.

13. Newnan, D. G., Engineering Economic Analysis, pp. 53, 291-292, Engineering Press, 1976.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 69 Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93940	2
4.	Professor G. N. Vanderplaats, Code 69Vd Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93940	1
5.	Professor M. D. Kelleher, Code 69Kk Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93940	1
6.	LT Walter B. Cole, USN 1409 Greeley Court Virginia Beach, Virginia 23456	1